```
ASM80 GETF.ASM DEBUG


ISIS-II 8080/8085 MACRO ASSEMBLER, V2.0        GETF     PAGE    1
GETF  - CP/M 3.0 Input Redirection - August 1982

  LOC  OBJ        SEQ        SOURCE STATEMENT

                    1 $title('GETF  - CP/M 3.0 Input Redirection - August 1982')
                    2         name    getf
                    3 ;*****************************************************************
                    4 ;
                    5 ;       get  'Input Redirection Initializer'  version 3.0
                    6 ;
                    7 ;       11/30/82 - Doug Huskey
                    8 ;*****************************************************************
                    9 ;
                   10 ;
                   11 ;       Copyright (c) 1982
                   12 ;       Digital Research
                   13 ;       P.O. Box 579
                   14 ;       Pacific Grove, Ca.
                   15 ;       93950
                   16 ;
                   17 ;
                   18 ;       generation procedure
                   19 ;
                   20 ;       seteof get.plm
                   21 ;       seteof getscan.dcl
                   22 ;       seteof getf.asm
                   23 ;       seteof getscan.plm
                   24 ;       seteof parse.asm
                   25 ;       is14
                   26 ;       asm80 getf.asm debug
                   27 ;       asm80 mcd80a.asm debug
                   28 ;       asm80 parse.asm debug
                   29 ;       plm80 get.plm pagewidth(100) debug optimize
                   30 ;       link mcd80a.obj,get.obj,parse.obj,getf.obj,plm80.lib to get.mod
                   31 ;       locate get.mod code(0100H) stacksize(100)
                   32 ;       era get.mod
                   33 ;       cpm
                   34 ;       objcpm get
                   35 ;       rmac getrsx
                   36 ;       link getrsx[op]
                   37 ;       era get.rsx
                   38 ;       ren get.rsx=getrsx.prl
                   39 ;       gencom get.com
                   40 ;       gencom get.com get.rsx
                   41 ;
                   42 ;
                   43 ;
                   44 ;       This module is called as an external routine by the
                   45 ;       PL/M routines GET and SUBMIT.  It is passed a structure
                   46 ;       with the following format:
                   47 ;
                   48 ;
                   49 ;                 declare getpb structure
                   50 ;                   (input$type   byte,
                   51 ;                    echo$flag    byte,
                   52 ;                    filtered$flag byte,
```


cp/m plus Vu3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. # CP3-135

```
 LOC  OBJ        SEQ        SOURCE STATEMENT

                   53 ;                   program$flag  byte);
                   54 ;
                   55 ;       input$type      = 0     > console input (default)
                   56 ;                       = 1     > auxiliary output
                   57 ;
                   58 ;       echo            = true  > echo input to real device
                   59 ;                                 (default)
                   60 ;                       = false > don't echo input (output is
                   61 ;                                 still echoed)
                   62 ;       filtered        = true  > convert control characters
                   63 ;                                 to a printable form
                   64 ;                                 preceeded by an ^ in echo
                   65 ;                                 (default)
                   66 ;                       = false > no character conversions
                   67 ;       program         = false > continue until EOF or
                   68 ;                                 GET INPUT FROM CONSOLE
                   69 ;                                 command
                   70 ;                       = true  > active only until program
                   71 ;                                 termination
                   72 ;
                   73         public  getf
                   74         extrn   mon1,fcb,memsiz
                   75 ;
                   76 ;
FFFF               77 true            equ     0ffffh
0000               78 false           equ     00000h
                   79 ;
FFFF               80 biosfunctions   equ     true            ;intercept BIOS conin & constat
                   81 ;
                   82 ;
                   83 ;       low memory locations
                   84 ;
0000               85 wboot   equ     0000h
0001               86 wboota  equ     wboot+1
                   87 ;
                   88 ;       equates for non graphic characters
                   89 ;
000D               90 cr      equ     0dh     ; carriage return
000A               91 lf      equ     0ah     ; line feed
                   92 ;
                   93 ;       BDOS function equates
                   94 ;
0001               95 cinf    equ     1       ;read character
0002               96 coutf   equ     2       ;output character
0006               97 crawf   equ     6       ;raw console I/O
000A               98 creadf  equ     10      ;read buffer
000B               99 cstatf  equ     11      ;status
0005              100 pchrf   equ     5       ;print character
0009              101 pbuff   equ     9       ;print buffer
000F              102 openf   equ     15      ;open file
0010              103 closef  equ     16      ;close file
0013              104 delf    equ     19      ;delete file
0014              105 dreadf  equ     20      ;disk read
001A              106 dmaf    equ     26      ;set dma function
0019              107 curdrv  equ     25
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
LOC  OBJ          SEQ         SOURCE STATEMENT

0020               108 userf   equ     32      ;set/get user number
0031               109 scbf    equ     49      ;set/get system control block word
003C               110 rsxf    equ     60      ;RSX function call
0080               111 initf   equ     128     ;GET initialization sub-function no.
0081               112 Killf   equ     129     ;GET delete sub-function no.
008D               113 jKillf  equ     141     ;JOURNAL delete sub-function no.
                   114 ;
                   115 ;       System Control Block definitions
                   116 ;
003A               117 scba    equ     03ah    ;offset of scbadr from SCB base
00B4               118 ccpflg2 equ     0b4h    ;offset of 2nd ccp flag byte from pg bound
00AA               119 errflg  equ     0aah    ;offset of error flag from page boundary
00CF               120 conmode equ     0cfh    ;offset of console mode from page boundary
00D4               121 listcp  equ     0d4h    ;offset of ^P flag from page boundary
00F9               122 common  equ     0f9h    ;offset of common memory base from pg. bound
0068               123 wbootfx equ     068h    ;offset of warm boot jmp from page. bound
006E               124 constfx equ     06eh    ;offset of constat jmp from page. bound
0074               125 coninfx equ     074h    ;offset of conin jmp from page. bound
007A               126 conoufx equ     07ah    ;offset of conout jmp from page. bound
0080               127 listfx  equ     080h    ;offset of list jmp from page. bound
0098               128 realdos equ     098h    ;offset of real BDOS entry from pg. bound
                   129 ;
                   130 ;       Restore mode equates    (used with inr a, rz, rm, rpe, ret)
                   131 ;
00FF               132 norestore       equ     0ffh    ;no BIOS interception
007F               133 biosonly        equ     07fh    ;restore BIOS jump table only
0080               134 stfix   equ     080h    ;restore BIOS jump table and
                   135                                 ;restore JMP in RESBDOS for constat
0000               136 everything      equ     0       ;restore BIOS jump table and jmps in
                   137                                 ;RESBDOS (default mode)
                   138 ;
                   139 ;       Instructions
                   140 ;
0021               141 lxih            equ     21h     ;LXI H, instruction
00C3               142 jmpi            equ     0c3h    ;JMP instruction
0022               143 shldi           equ     22h     ;SHLD instruction
                   144 ;
                   145 ;***************************************************************
                   146 ;               START OF INITIALIZATION CODE
                   147 ;***************************************************************
                   148
                   149                 cseg
                   150
                   151 getf:
                   152         ;get parameters
0000 60            153         mov     h,b
0001 69            154         mov     l,c             ;HL = .(parameter block)
0002 7E            155         mov     a,m             ;input type 0=con:,1=aux:
0003 FE01          156         cpi     1               ;is it aux?
0005 CA2301    C   157         jz      notimp          ;error if so
0008 23            158         inx     h
0009 7E            159         mov     a,m             ;echo/noecho mode
000A 32ED01    C   160         sta     echo
000D 23            161         inx     h
000E 7E            162         mov     a,m             ;cooked/raw mode
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
LOC  OBJ        SEQ        SOURCE STATEMENT

000F 32EE01   C  163        sta     cooked
0012 23          164        inx     h
0013 7E          165        mov     a,m
0014 32EF01   C  166        sta     program
                 167        ;
                 168        ;check if enough memory
                 169        ;
0017 2A0000   E  170        lhld    memsiz
001A 7C          171        mov     a,h
001B FE20        172        cpi     20h
001D DA3E01   C  173        jc      nomem
                 174        ;
                 175        ;close to get those blocks in the directory
                 176        ;
0020 110000   E  177        lxi     d,fcb
0023 0E10        178        mvi     c,closef
0025 CD0000   E  179        call    bdos
                 180        ;
                 181        ;check if drive specified
0028 210000   E  182        lxi     h,fcb
002B 7E          183        mov     a,m             ;drive code
002C B7          184        ora     a               ;default?
002D C23900   C  185        jnz     movfcb
                 186        ;
                 187        ;set to current drive, if  not
                 188        ;
0030 E5          189        push    h               ;save .fcb
0031 0E19        190        mvi     c,curdrv
0033 CD0000   E  191        call    bdos
0036 E1          192        pop     h               ;a=current drive, hl=.fcb
0037 3C          193        inr     a
0038 77          194        mov     m,a             ;set fcb to force drive select
                 195        ;
                 196 movfcb: ;copy default fcb up into data area for move to RSX
                 197        ;
0039 11F101   C  198        lxi     d,subfcb
003C 012000      199        lxi     b,32            ;length of fcb
003F CD5601   C  200        call    ldir            ;move it to subfcb
                 201        ;
                 202        ;initialize other variables to be moved to RSX
                 203        ;
0042 CD4401   C  204        call    getusr          ;get current user number
0045 32F001   C  205        sta     subusr          ;save for redirection file I/O
0048 CD4C01   C  206        call    getscbadr
004B 22DB01   C  207        shld    scbadr          ;System Control Block address
                 208        ;
                 209        ;get real BDOS address (bypass chain to check for user break)
                 210        ;
004E 2E98        211        mvi     l,realdos
0050 5E          212        mov     e,m
0051 23          213        inx     h
0052 56          214        mov     d,m
0053 EB          215        xchg
0054 22EB01   C  216        shld    realbdos+1
                 217        ;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
LOC  OBJ          SEQ         SOURCE STATEMENT

                  218         ;check for user abort
                  219         ;
0057 EB           220         xchg
0058 2ECF         221         mvi     l,conmode
005A 7E           222         mov     a,m
005B F601         223         ori     1               ;set ^C status mode
005D 77           224         mov     m,a
005E 0E0B         225         mvi     c,cstatf
0060 CDEA01    C  226         call    realbdos        ;check for user abort
0063 B7           227         ora     a
0064 C22B01    C  228         jnz     error1          ;abort if so
                  229         ;
                  230         ;get address of initialization table in RSX
                  231         ;
0067 0E3C         232         mvi     c,rsxf
0069 116101    C  233         lxi     d,journkill
006C CD0000    E  234         call    bon1            ;terminate any PUT INPUT commands
006F 0E3C         235         mvi     c,rsxf
0071 116201    C  236         lxi     d,rsxinit
0074 CD0000    E  237         call    bon1            ;call GET.RSX initialization routine
0077 E5           238         push    h               ;save for move at end of setup
0078 5E           239         mov     e,m
0079 23           240         inx     h
007A 56           241         mov     d,m             ;DE = .RSXKILL flag
007B D5           242         push    d               ;set flag to zero if successfull
007C 23           243         inx     h               ;HL = .(real bios status routine)
007D E5           244         push    h
                  245         ;
                  246 if biosfunctions
                  247         ;
                  248         ;check if BIOS jump table looks valid (jmp in right places)
007E 2A0100       249         lhld    wboota
0081 110300       250         lxi     d,3
0084 19           251         dad     d               ;HL = .(jmp constat address)
0085 7E           252         mov     a,m
0086 FEC3         253         cpi     jmpi            ;should be a jump
0088 C21201    C  254         jnz     bioserr         ;skip bios redirection if not
008B 19           255         dad     d               ;HL = .(jmp conin address)
008C 7E           256         mov     a,m
008D FEC3         257         cpi     jmpi
008F C21201    C  258         jnz     bioserr         ;skip bios redirection if not
                  259         ;
                  260         ;fix up RESBDOS to do BIOS calls to intercepted functions
                  261         ;
0092 2ADB01    C  262         lhld    scbadr
0095 2EFA         263         mvi     l,common+1
0097 7E           264         mov     a,m             ;get high byte of common base
0098 B7           265         ora     a
0099 C2A400    C  266         jnz     fix0            ;high byte = zero if non-banked
009C 3E7F         267         mvi     a,biosonly
009E 32E101    C  268         sta     biosmode
00A1 C3D000    C  269         jmp     trap            ;skip code that fixes resbdos
                  270         ;fix BIOS constat
00A4 2E6E         271 fix0:   mvi     l,constfx               ;hl = .constfx in SCB
00A6 7E           272         mov     a,m
```

```
LOC  OBJ        SEQ        SOURCE STATEMENT

00A7 FEC3        273        cpi     jmpi            ;is it a jump instruction?
00A9 CAB100   C  274        jz      fix1            ;jump if so
00AC 3E7F        275        mvi     a,biosonly      ;whoops already changed
00AE 32E101   C  276        sta     biosmode        ;restore jump table only
00B1 3621        277 fix1:  mvi     m,lxih
                 278        ;fix BIOS conin
00B3 2E74        279        mvi     l,coninfx       ;hl = .coninfx in SCB
00B5 7E          280        mov     a,m
00B6 FEC3        281        cpi     jmpi            ;is it a jump instruction?
00B8 3AE101   C  282        lda     biosmode
00BB CAC400   C  283        jz      fix2            ;jump if so
00BE FE7F        284        cpi     biosonly
00C0 C21201   C  285        jnz     bioserr         ;error if conin is LXI but not constat
00C3 AF          286        xra     a               ;zero accumulator to jnz below
00C4 FE7F        287 fix2:  cpi     biosonly        ;was const already an LXI h?
00C6 C2CE00   C  288        jnz     fix3            ;jmp if not
00C9 3E80        289        mvi     a,stfix ;restore constat jmp but not conin
00CB 32E101   C  290        sta     biosmode
00CE 3621        291 fix3:  mvi     m,lxih
                 292        ;get addresses of RSX const and conin traps
00D0 E1          293 trap:  pop     h
00D1 4E          294        mov     c,m             ;HL = .(.bios constat trap)
00D2 23          295        inx     h
00D3 46          296        mov     b,m             ;BC = .bios constat trap in RSX
00D4 23          297        inx     h
00D5 E5          298        push    h               ;save for CONIN setup
                 299        ;
                 300        ;patch RSX constat entry into BIOS jump table
                 301        ;save real constat address in RSX exit table
                 302        ;
00D6 2A0100      303        lhld    wboota
00D9 110400      304        lxi     d,4
00DC 19          305        dad     d               ;HL = .(.jmp constat address)
00DD 22E301   C  306        shld    const.jmp       ;save for RSX restore at end
00E0 5E          307        mov     e,m
00E1 71          308        mov     m,c
00E2 23          309        inx     h
00E3 56          310        mov     d,m             ;DE = constat address
00E4 70          311        mov     m,b             ;BIOS constat jumps to RSX
00E5 EB          312        xchg
00E6 22DD01   C  313        shld    biosta          ;save real constat address
                 314        ;
                 315        ;get address of RSX bios conin entry point
                 316        ;
00E9 E1          317        pop     h               ;HL = .(RSX BIOS conin trap)
00EA 4E          318        mov     c,m
00EB 23          319        inx     h
00EC 46          320        mov     b,m
                 321        ;
                 322        ;patch RSX conin entry into BIOS jump table
                 323        ;save real conin address in RSX exit table
                 324        ;
00ED EB          325        xchg
00EE 23          326        inx     h               ;past jmp instruction
00EF 23          327        inx     h               ;HL = .(conin address)
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
LOC  OBJ          SEQ         SOURCE STATEMENT

00F0 22E701   C   328         shld    coninjmp
00F3 5E           329         mov     e,m
00F4 71           330         mov     m,c
00F5 23           331         inx     h
00F6 56           332         mov     d,m             ;DE = conin address
00F7 70           333         mov     m,b             ;BIOS conin jumps to RSX
00F8 EB           334         xchg
00F9 22DF01   C   335         shld    biosin          ;save real conin address
                  336 endif
                  337         ;
                  338         ;move data area to RSX
                  339         ;
                  340 rsxmov:
00FC E1           341         pop     h               ;HL = .Kill flag in RSX
00FD 34           342         inr     m               ;switch from FF to 0
00FE 21DB01   C   343         lxi     h,movstart
0101 D1           344         pop     d               ;RSX data area address
0102 013700       345         lxi     b,movend-movstart
0105 CD5601   C   346         call    ldir
0108 0E06         347         mvi     c,crawf
010A 1EFD         348         mvi     e,0fdh          ;raw console input
010C CD0000   E   349         call    mon1            ;prime RSX by reading a char
010F C30000       350         jmp     wboot
                  351
                  352 if biosfunctions
                  353 ;
                  354 ;       can't do BIOS redirection
                  355 ;
                  356 bioserr:
0112 116301   C   357         lxi     d,nobios
0115 0E09         358         mvi     c,pbuff
0117 CD0000   E   359         call    mon1
011A 21E101   C   360         lxi     h,biosmode
011D 36FF         361         mvi     m,norestore     ;no bios redirection
011F E1           362         pop     h               ;throw away bios constat trap adr
0120 C3FC00   C   363         jmp     rsxmov
                  364 endif
                  365 ;
                  366 ;       auxiliary redirection
                  367 ;
                  368 notimp:
0123 118801   C   369         lxi     d,notdone
                  370 error:
0126 0E09         371         mvi     c,pbuff
0128 CD0000   E   372         call    mon1
012B 0E10         373 error1: mvi     c,closef
012D 110000   E   374         lxi     d,fcb
0130 CD0000   E   375         call    mon1
0133 0E13         376         mvi     c,delf
0135 110000   E   377         lxi     d,fcb
0138 CD0000   E   378         call    mon1
013B C30000       379         jmp     wboot
                  380 ;
                  381 ;       insufficient memory
                  382 ;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
LOC  OBJ          SEQ         SOURCE STATEMENT

013E 11BE01   C   383 nomem:  lxi     d,memerr
0141 C32601   C   384         jmp     error
                  385
                  386 ;
                  387 ;       get/set user number
                  388 ;
0144 3EFF         389 getusr: mvi     a,0ffh          ;get current user number
0146 5F           390 setusr: mov     e,a             ;set current user number (in A)
0147 0E20         391         mvi     c,userf
0149 C30000   E   392         jmp     mon1
                  393 ;
                  394 ;       get system control block address
                  395 ;       (BDOS function #49)
                  396 ;
                  397 ;       exit:   hl = system control block address
                  398 ;
                  399 getscbadr:
014C 0E31         400         mvi     c,scbf
014E 115401   C   401         lxi     d,data49
0151 C30000   E   402         jmp     mon1
                  403 ;
0154 3A           404 data49: db      scba,0          ;data structure for getscbadd
0155 00
                  405 ;
                  406 ;       copy memory bytes (emulates z80 ldir instruction)
                  407 ;
0156 7E           408 ldir:   mov     a,m             ;get byte
0157 12           409         stax    d               ;store it at destination
0158 23           410         inx     h               ;advance pointers
0159 13           411         inx     d
015A 0B           412         dcx     b               ;decrement byte count
015B 79           413         mov     a,c             ;loop if non-zero
015C B0           414         ora     b
015D C25601   C   415         jnz     ldir
0160 C9           416         ret
                  417 ;
                  418 ;***************************************************************
                  419 ;               DATA AREA
                  420 ;***************************************************************
                  421
                  422         ;
0161 8D           423 journkill:      db      jkillf
0162 80           424 rsxinit:        db      initf
0163 5741524E     425 nobios:         db      'WARNING: Connot redirect from BIOS',cr,lf,'$'
0167 494E473A
016B 2043616E
016F 6E6F7420
0173 72656469
0177 72656374
017B 2066726F
017F 6D204249
0183 4F53
0185 0D
0186 0A
0187 24
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
LOC  OBJ        SEQ        SOURCE STATEMENT

                426 notdone:
0188 4552524F   427        db 'ERROR: Auxiliary device redirection not implemented',cr,lf,'$'
018C 523A2041
0190 7578696C
0194 69617279
0198 20646576
019C 69636520
01A0 72656469
01A4 72656374
01A8 696F6E20
01AC 6E6F7420
01B0 696D706C
01B4 656D656E
01B8 746564
01BB 0D
01BC 0A
01BD 24
                428 memerr:
01BE 4552524F   429        db 'ERROR: Insufficient Memory',cr,lf,'$'
01C2 523A2049
01C6 6E737566
01CA 66696369
01CE 656E7420
01D2 4D656D6F
01D6 7279
01D8 0D
01D9 0A
01DA 24
                430        ;
                431 ;**************************************************************
                432 ;      Following variables are initialized by GET.COM
                433 ;      and moved to the GET RSX - Their order must not be changed
                434 ;**************************************************************
                435        ;
                436        ;
                437        ;
                438 movstart:
                439 inittable:                    ;addresses used by GET.COM for
01DB 0000       440 scbadr: dw     0             ;address of System Control Block
                441        ;
                442        if biosfunctions      ;GET.RSX initialization
                443        ;
01DD 0000       444 biosta: dw     0             ;set to real BIOS routine
01DF 0000       445 biosin: dw     0             ;set to real BIOS routine
                446        ;
                447                              ;restore only if changed when removed.
                448 biosmode:
01E1 00         449        db     0             ;if non-zero change LXI @jmpadr to JMP
                450                              ;when removed.
                451 restorebios:
                452        ;hl = real constat routine
                453        ;de = real conin routine
01E2 22         454        db     shldi
                455 const.jmp:
01E3 0000       456        dw     0             ;address of const jmp initialized by COM
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 LOC  OBJ        SEQ        SOURCE STATEMENT

 01E5 EB          457        xchg
 01E6 22          458        db      shldi
                  459 conin.jmp:
 01E7 0000        460        dw      0            ;address of conin jmp initialized by CCM
 01E9 C9          461        ret
                  462        endif
                  463        ;
                  464 realbdos:
 01EA C30000      465        jmp     0            ;address filled in by CCM
                  466        ;
 01ED 01          467 echo:  db      1
 01EE 00          468 cooked: db     0
                  469        ;
                  470 program:
 01EF 00          471        db      0            ;true if only program input
 01F0 00          472 subusr: db     0            ;user number for redirection file
 01F1 01          473 subfcb: db     1            ;a:
 01F2 53595349    474        db      'SYSIN   '
 01F6 4E202020
 01FA 535542      475        db      'SUB'
 01FD 00          476        db      0,0
 01FE 00
 01FF 00          477 submod: db     0
 0200 00          478 subrc: db      0
 0010             479        ds      16           ;map
 0211 00          480 subcr: db      0
                  481        ;
                  482 movend:
                  483 ;*****************************************************************
                  484        end
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
PUBLIC SYMBOLS
GETF   C 0000

EXTERNAL SYMBOLS
FCB    E 0000    MEMSIZ E 0000    MON1   E 0000

USER SYMBOLS
BIOSER C 0112    BIOSFU A FFFF    BIOSIN C 01DF    BIOSMO C 01E1    BIOSON A 007F    BIOSTA C 01DB    CCPFLG A 00B4
CINF   A 0001    CLOSEF A 0010    COMMON A 00F9    CONINF A 0074    CONINJ C 01E7    CONMOD A 00CF    CONOUF A 007A
CONSTF A 006E    CONSTJ C 01E3    COOKED C 01EE    COUTF  A 0002    CR     A 000D    CRAWF  A 0006    CREADF A 000A
CSTATF A 000B    CURDRV A 0019    DATA49 C 0154    DELF   A 0013    DMAF   A 001A    DREADF A 0014    ECHO   C 01ED
ERRFLG A 00AA    ERROR  C 0126    ERROR1 C 012B    EVERYT A 0000    FALSE  A 0000    FCB    E 0000    FIX0   C 00A4
FIX1   C 00B1    FIX2   C 00C4    FIX3   C 00CE    GETF   C 0000    GETSCB C 014C    GETUSR C 0144    INITF  A 0080
INITTA C 01DB    JKILLF A 008D    JMPI   A 00C3    JOURNK C 0161    KILLF  A 0081    LDIR   C 0156    LF     A 000A
LISTCP A 00D4    LISTFX A 0080    LXIH   A 0021    MEMERR C 01BE    MEMSIZ E 0000    MON1   E 0000    MOVEND C 0212
MOVFCB C 0039    MOVSTA C 01DB    NOBIOS C 0163    NOMEM  C 013E    NOREST A 00FF    NOTDON C 0188    NOTIMP C 0123
OPENF  A 000F    PBUFF  A 0009    PCHRF  A 0005    PROGRA C 01EF    REALBD C 01EA    REALDO A 0098    RESTOR C 01E2
RSXF   A 003C    RSXINI C 0162    RSXMOV C 00FC    SCBA   A 003A    SCBADR C 01DB    SCBF   A 0031    SETUSR C 0146
SHLDI  A 0022    STFIX  A 0080    SUBCR  C 0211    SUBFCB C 01F1    SUBMOD C 01FF    SUBRC  C 0200    SUBUSR C 01F0
TRAP   C 00D0    TRUE   A FFFF    USERF  A 0020    WBOOT  A 0000    WBOOTA A 0001    WBOOTF A 0068

ASSEMBLY COMPLETE,  NO ERRORS
```

```
ISIS-II PL/M-80 V3.1 COMPILATION OF MODULE GET
OBJECT MODULE PLACED IN GET.OBJ
COMPILER INVOKED BY:  PLM80 GET.PLM XREF PAGEWIDTH(100) DEBUG OPTIMIZE
```

```
          $ TITLE('CP/M 3.0 --- GET user interface')
  1       get:
          do;

          /*
            Copyright (C) 1982
            Digital Research
            P.O. Box 579
            Pacific Grove, CA 93950
          */

          /*
          Written:  30 July 82  by John Knight
                    12 Sept 82  by Doug Huskey
          */

          /*****************************************
          *                                       *
          *      LITERALS AND GLOBAL VARIABLES    *
          *                       *
          *****************************************/

  2  1    declare
             true           literally '1',
             false          literally '0',
             forever        literally 'while true',
             lit            literally 'literally',
             proc           literally 'procedure',
             dcl            literally 'declare',
             addr           literally 'address',
             cr             literally '13',
             lf             literally '10',
             ctrlc          literally '3',
             ctrlx          literally '18h',
             bksp           literally '8',
             con$type       literally '0',
             aux$type       literally '1',
             con$width$offset    literally '1ah',
             ccp$flag$offset literally '18h',
             get$rsx$init       literally '128',
             get$rsx$kill       literally '129',
             get$rsx$fcb        literally '130',
             cpmversion     literally '30h';

  3  1      declare ccp$flag byte;
  4  1      declare con$width byte;
  5  1      declare i byte;
  6  1      declare begin$buffer address;
  7  1      declare buf$length byte;
  8  1      declare no$chars byte;
  9  1      declare get$init$pb byte initial(get$rsx$init);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
10  1       declare get$kill$pb byte initial(get$rsx$kill);
11  1       declare get$fcb$pb byte initial(get$rsx$fcb);
12  1       declare input$type byte;

13  1       declare
              sub$fcb (*)   byte data (0,'SYSIN   $$$'),
              get$msg (*)   byte data ('Getting console input from $');

            /* scanner variables and data */
14  1       declare
              options(*) byte data
                  ('INPUT"FROM"FILE"STATUS"CONDITIONAL"',
                   'FALSE"TRUE"CONSOLE"CONIN:"AUXILIARY"',
                   'AUXIN:"END"CON:"AUX:"NOT"ECHO"FILTERED"SYSTEM"PROGRAM',0FFH),

              options$offset(*) byte data
                  (0,6,11,16,23,35,41,46,54,61,71,78,82,87,92,96,101,110,117,124),

              end$list    byte data (0ffh),

              delimiters(*) byte data (0,'[]=, ./;',0,0ffh),

              SPACE   byte data(5),

              buf$ptr address,
              index   byte,
              endbuf  byte,
              j       byte initial(0),
              delimiter   byte;

15  1           declare end$of$string byte initial ('"');

16  1       declare getpb structure
              (input$type   byte,
               echo$flag    byte,
               filtered$flag byte,
               program$flag  byte)
              initial(con$type,true,true,true);

17  1       declare scbpd structure
              (offset byte,
               set    byte,
               value  address);

18  1       declare parse$fn structure
              (buff$adr address,
               fcb$adr  address);

19  1      declare plm label public;

            /**************************************
             *                                    *
             *      B D O S   INTERFACE           *
             *                                    *
             **************************************/
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
20  1       mon1:
              procedure (func,info) external;
21  2           declare func byte;
22  2           declare info address;
23  2         end mon1;

24  1       mon2:
              procedure (func,info) byte external;
25  2           declare func byte;
26  2           declare info address;
27  2         end mon2;

28  1       mon3:
              procedure (func,info) address external;
29  2           declare func byte;
30  2           declare info address;
31  2         end mon3;

32  1       declare cmdrv     byte    external;  /* command drive      */
33  1       declare fcb (1)   byte    external;  /* 1st default fcb    */
34  1       declare fcb16 (1) byte    external;  /* 2nd default fcb    */
35  1       declare pass0     address external;  /* 1st password ptr   */
36  1       declare len0      byte    external;  /* 1st passwd length  */
37  1       declare pass1     address external;  /* 2nd password ptr   */
38  1       declare len1      byte    external;  /* 2nd passwd length  */
39  1       declare tbuff (1) byte    external;  /* default dma buffer */

        /**************************************
         *                                    *
         *      B D O S   Externals           *
         *                                    *
         **************************************/

40  1       printchar:
              procedure(char);
41  2           declare char byte;
42  2           call mon1(2,char);
43  2         end printchar;

44  1       conin:
              procedure byte;
45  2           return mon2(6,0fdh);
46  2         end conin;

47  1       print$buf:
              procedure (buffer$address);
48  2           declare buffer$address address;
49  2           call mon1 (9,buffer$address);
50  2         end print$buf;

51  1       read$console$buf:
              procedure (buffer$address,max) byte;
52  2           declare buffer$address address;
53  2           declare new$max based buffer$address address;
54  2           declare max byte;
55  2           new$max = max;
56  2           call mon1(10,buffer$address);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
57  2         buffer$address = buffer$address + 1;
58  2         return new$max; /* actually number of characters input */
59  2       end read$console$buf;

60  1       version: procedure address;
              /* returns current cp/m version # */
61  2         return mon3(12,0);
62  2         end version;

63  1       check$con$stat: procedure byte;
64  2         return mon2(11,0);
65  2       end check$con$stat;

66  1       open$file:
            procedure (fcb$address) address;
67  2         declare fcb$address address;
68  2         return mon3(15,fcb$address);
69  2       end open$file;

70  1       set$dma: procedure(dma);
71  2         declare dma address;
72  2         call mon1(26,dma);
73  2       end set$dma;

            /* 0ffh ==> return BDOS errors */
74  1       return$errors: procedure (mode);
75  2         declare mode byte;
76  2         call mon1(45,mode);
77  2       end return$errors;

78  1       getscbbyte: procedure (offset) byte;
79  2         declare offset byte;
80  2         scbpd.offset = offset;
81  2         scbpd.set = 0;
82  2         return mon2(49,.scbpd);
83  2       end getscbbyte;

84  1       setscbbyte:
            procedure (offset,value);
85  2         declare offset byte;
86  2         declare value byte;
87  2         scbpd.offset = offset;
88  2         scbpd.set = 0ffh;
89  2         scbpd.value = double(value);
90  2         call mon1(49,.scbpd);
91  2       end setscbbyte;

92  1     get$console$mode: procedure address;
          /* returns console mode */
93  2       return mon3(6dh,0ffffh);
94  2     end get$console$mode;

95  1     set$console$mode: procedure (new$value);
96  2       declare new$value address;
97  2       call mon1(6dh,new$value);
98  2     end set$console$mode;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
 99  1     rsx$call: procedure (rsxpb) address;
           /* call Resident System Extension */
100  2       declare rsxpb address;
101  2       return mon3(60,rsxpb);
102  2     end rsx$call;

103  1     parse: procedure (pfcb) address external;
104  2       declare pfcb address;
105  2     end parse;

106  1     getf: procedure (input$type) external;
107  2           declare input$type address;
108  2     end getf;

        /**************************************
         *                                    *
         *      S U B R O U T I N E S         *
         *                                    *
         **************************************/

        /* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *

                          * * *  Option scanner  * * *

         * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */


109  1     separator: procedure(character) byte;

                          /* determines if character is a
                             delimiter and which one */
110  2         declare k   byte,
                   character   byte;

111  2         k = 1;
112  2     loop:   if delimiters(k) = end$list then return(0);
114  2         if delimiters(k) = character then return(k);    /* null = 25 */
116  2             k = k + 1;
117  2             go to loop;

118  2     end separator;

119  1     opt$scanner:    procedure(list$ptr,off$ptr,idx$ptr);
                          /* scans the list pointed at by idxptr
                             for any strings that are in the
                             list pointed at by list$ptr.
                             Offptr points at an array that
                             contains the indices for the known
                             list. Idxptr points at the index
                             into the list. If the input string
                             is unrecognizable then the index is
                             0, otherwise > 0.

                          First, find the string in the known
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        list that starts with the same first
                        character.  Compare up until the next
                        delimiter on the input. if every input
                        character matches then check for
                        uniqueness.  Otherwise try to find
                        another known string that has its first
                        character match, and repeat.  If none
                        can be found then return invalid.

                        To test for uniqueness, start at the
                        next string in the Knwon list and try
                        to get another match with the input.
                        If there is a match then return invalid.

                        else move pointer past delimiter and
                        return.

                    P.Balma     */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
120  2      declare
                buff        based buf$ptr (1) byte,
                idx$ptr     address,
                off$ptr     address,
                list$ptr    address;

121  2      declare
                i       byte,
                j       byte,
                list        based list$ptr (1) byte,
                offsets     based off$ptr (1) byte,
                wrd$pos     byte,
                character   byte,
                letter$in$word  byte,
                found$first byte,
                start       byte,
                index       based idx$ptr byte,
                save$index  byte,
                (len$new,len$found) byte,
                valid       byte;

        /*****************************************************************************/
        /*          internal subroutines                                             */
        /*****************************************************************************/

122  2      check$in$list: procedure;
                    /* find known string that has a match with
                       input on the first character.  Set index
                       = invalid if none found.   */

123  3          declare i   byte;

124  3          i = start;
125  3          wrd$pos = offsets(i);
126  3          do while list(wrd$pos) <> end$list;
127  4              i = i + 1;
128  4              index = i;
129  4              if list(wrd$pos) = character then return;
```

```
131  4            wrd$pos = offsets(i);
132  4        end;
                     /* could not find character */
133  3        index = 0;
134  3        return;
135  3    end check$in$list;

136  2    setup:  procedure;
137  3        character = buff(0);
138  3        call check$in$list;
139  3        letter$in$word = wrd$pos;
                     /* even though no match may have occurred, position
                        to next input character.  */
140  3        i = 1;
141  3        character = buff(1);
142  3    end setup;

143  2    test$letter:    procedure;
                     /* test each letter in input and known string */

144  3        letter$in$word = letter$in$word + 1;

                          /* too many chars input? 0 means
                             past end of known string */
145  3        if list(letter$in$word) = end$of$string then valid = false;
             else
147  3        if list(letter$in$word) <> character then valid = false;

             i = i + 1;
150  3        character = buff(i);

151  3    end test$letter;

152  2    skip:   procedure;
                          /* scan past the offending string;
                             position buf$ptr to next string...
                             skip entire offending string;
                             ie., falseopt=mod, [note: comma or
                             space is considered to be group
                             delimiter] */
153  3        character = buff(i);
154  3        delimiter = separator(character);
             /* No skip for GET */
155  3            do while ((delimiter < 1) or (delimiter > 9));
156  4            i = i + 1;
157  4            character = buff(i);
158  4            delimiter = separator(character);
159  4        end;
160  3        endbuf = i;
161  3        buf$ptr = buf$ptr + endbuf + 1;
162  3        return;
163  3    end skip;

164  2    eat$blanks: procedure;

165  3        declare charac  based buf$ptr byte;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
166  3        do while ((delimiter != separator(charac)) = SPACE);
167  4           buf$ptr = buf$ptr + 1;
168  4        end;

169  3     end eat$blanks;

        /*****************************************************************************/
        /*         end of internals                    */
        /*****************************************************************************/


                          /* start of procedure */
170  2        if delimiter = 9 then
171  2           return;
172  2        call eat$blanks;
173  2        start = 0;
174  2        call setup;

                          /* match each character with the option
                             for as many chars as input
                             Please note that due to the array
                             indices being relative to 0 and the
                             use of index both as a validity flag
                             and as a index into the option/mods
                             list, index is forced to be +1 as an
                             index into array and 0 as a flag*/

175  2        do while index <> 0;
176  3           start = index;
177  3           delimiter = separator(character);

                          /* check up to input delimiter */

178  3           valid = true;       /* test$letter resets this */
179  3           do while delimiter = 0;
180  4              call test$letter;
181  4              if not valid then go to exit1;
183  4              delimiter = separator(character);
184  4           end;

185  3           go to good;

                          /* input ^= this known string;
                             get next known string that
                             matches */
186  3     exit1:      call setup;
187  3        end;
                          /* fell through from above, did
                             not find a good match*/
188  2        endbuf = i;         /* skip over string & return*/
189  2        call skip;
190  2        return;

                          /* is it a unique match in options
                             list? */
191  2     good:   endbuf = i;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
192  2          len$found = endbuf;
193  2          save$index = index;
194  2          valid = false;
195  2      next$opt:
                start = index;
196  2          call setup;
197  2          if index = 0 then go to finished;

                           /* look at other options and check
                              uniqueness */

199  2          len$new = offsets(index + 1) - offsets(index) - 1;
200  2          if len$new = len$found then do;
202  3              valid = true;
203  3              do j = 1 to len$found;
204  4                  call test$letter;
205  4                  if not valid then go to next$opt;
207  4              end;
208  3          end;
209  2          else go to nextopt;
                           /* fell through...found another valid
                              match --> ambiguous reference */
210  2          index = 0;
211  2          call skip;      /* skip input field to next delimiter*/
212  2          return;

213  2      finished:           /* unambiguous reference */
               index = save$index;
214  2          buf$ptr = buf$ptr + endbuf;
215  2          call eat$blanks;
216  2          if delimiter <> 0 then
217  2               buf$ptr = buf$ptr + 1;
                   else
218  2               delimiter = 5;
219  2          return;

220  2      end opt$scanner;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

221  1      crlf:  proc;
222  2          call printchar(cr);
223  2          call printchar(lf);
224  2          end crlf;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

           /* fill string @ s for c bytes with f */
225  1      fill: procedure(s,f,c);
226  2        declare s address;
227  2        declare (f,c) byte;
228  2        declare a based s byte;
229  2        do while (c:=c-1) <> 255;
230  3          a=f;
231  3          s=s+1;
232  3        end;
233  2      end fill;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

                /* The error processor.  This routine prints the command line
                  with a carot '^' under the offending delimiter, or sub-string.
                  The code passed to the routine determines the error message
                  to be printed beneath the command string.                    */

234   1         error: procedure (code);
235   2           declare (code,i,j,nlines,rem) byte;
236   2           declare (string$ptr,tstring$ptr) address;
237   2           declare chr1 based string$ptr byte;
238   2           declare chr2 based tstring$ptr byte;
239   2           declare carot$flag byte;

240   2         print$command: procedure (size);
241   3           declare size byte;
242   3           do j=1 to size;   /* print command string */
243   4             call printchar(chr1);
244   4             string$ptr = string$ptr + 1;
245   4           end;
246   3           call crlf;
247   3           do j=1 to size;   /* print carot if applicable */
248   4             if .chr2 = buf$ptr then do;
250   5               carot$flag = true;
251   5               call printchar('^');
252   5             end;
                    else
253   4               call printchar(' ');
254   4             tstring$ptr = tstring$ptr + 1;
255   4           end;
256   3           call crlf;
257   3         end print$command;

258   2           carot$flag = false;
259   2           string$ptr,tstring$ptr = begin$buffer;
260   2           con$width = getscbbyte(con$width$offset);
261   2           if con$width < 40 then con$width = 40;
263   2           nlines = buf$length / con$width;  /* num lines to print */
264   2           rem = buf$length mod con$width;   /* num extra chars to print */
265   2           if code <> 2 then do;
267   3             if ((code = 1) or (code = 4)) then  /* adjust carot pointer */
268   3               buf$ptr = buf$ptr - 1;    /* for delimiter errors */
269   3             else if code <> 5 then
270   3               buf$ptr = buf$ptr - endbuf - 1;   /* all other errors */
                  end;
272   2           call crlf;
273   2           do i=1 to nlines;
274   3             tstring$ptr = string$ptr;
275   3             call print$command(con$width);
276   3           end;
277   2           call print$command(rem);
278   2           if carot$flag then
279   2             call print$buf(.('Error at the ''^''; $'));
                  else
280   2             call print$buf(.('Error at end of line; $'));
281   2           if con$width < 65 then
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
282  2          call crlf;
283  2        do case code;
284  3          call print$buf(.('Invalid option or modifier$'));
285  3          call print$buf(.('End of line expected$'));
286  3          call print$buf(.('Invalid file specification$'));
287  3          call print$buf(.('Invalid command$'));
288  3          call print$buf(.('Invalid delimiter$'));
289  3          call print$buf(.('File not found$'));
290  3        end;
291  2        call crlf;
292  2        call mon1(0,0);
293  2      end error;

          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

294  1      ucase: procedure (char) byte;
295  2        declare char byte;
296  2        if char >= 'a' then
297  2          if char < '{' then
298  2            return (char-20h);
299  2        return char;
300  2      end ucase;

          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

301  1      getucase: procedure byte;
302  2        declare c byte;
303  2        c = ucase(conin);
304  2        return c;
305  2      end getucase;

          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

306  1      getpasswd: procedure;
307  2        declare (i,c) byte;
308  2        call crlf;
309  2        call crlf;
310  2        call print$buf(.('Enter Password: $'));
311  2      retry:
          call fill(.fcb16,' ',8);
312  2        do i=0 to 7;
313  3      nxtchr:
          if (c:=getucase) >= ' ' then
314  3          fcb16(i)=c;
315  3        if c = cr then
316  3          go to exit;
317  3        if c = ctrlx then
318  3          go to retry;
319  3        if c = bksp then do;
321  4          if i < 1 then
322  4            goto retry;
323  4          else do;
324  5            fcb16(i := i - 1) = ' ';
325  5            goto nxtchr;
326  5            end;
327  4          end;
328  3        if c = 3 then
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
329  3        call mon1(0,0);
330  3      end;
331  2    exit:
           c = check$con$stat;   /* clear raw i/o mode */
332  2    end getpasswd;

          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

333  1    print$fn: procedure (fcb$ad);
334  2      declare k byte;
335  2      declare fcb$ad address;
336  2      declare driv based fcb$ad byte;
337  2      declare fn based fcb$ad (12) byte;

338  2      call print$buf(.('file: $'));
339  2      if driv <> 0 then do;
341  3        call printchar('@'+driv);
342  3        call printchar(':');
343  3      end;
344  2      do k=1 to 11;
345  3        if k=9 then
346  3          call printchar(',');
347  3        if fn(k) <> ' ' then
348  3          call printchar(fn(k) and 07fh);
349  3      end;
350  2    end print$fn;

          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

351  1    try$open: procedure;
352  2      declare (error$code,a) address;
353  2      declare prog$flag based a byte;
354  2      declare code byte;


355  2      error$code = rsx$call(.get$fcb$pb);
356  2      if error$code <> 0ffh then do; /* 0ffh means no active get */
358  3        a = error$code - 2;
359  3        if prog$flag then           /* program input only? */
360  3          error$code = rsx$call(.get$kill$pb); /* kill if so */
361  3      end;
362  2      call setdma(.fcb16);  /* set dma to password */
363  2      call return$errors(0ffh);
364  2      error$code = open$file(.fcb);
365  2      call return$errors(0);
366  2      if low(error$code) = 0ffh then
367  2        if (code := high(error$code)) <> 0 then do;
369  3          if code = 7 then do;
371  4            call getpasswd;
372  4            call crlf;
373  4            call setdma(.fcb16);
374  4          end;
375  3          error$code=open$file(.fcb);
376  3        end;
377  2        else do;
378  3          buf$ptr = parse$fn.buff$adr;  /* adjust pointer to file */
379  3          call error(5);    /* file not found */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
380  3          end;
381  2        call print$buf(.get$msg);
382  2        if getscbbyte(26) < 48 then
383  2          call crlf;              /* console width */
384  2        call print$fn(.fcb);
385  2        call getf(.getpb);
386  2      end try$open;

         /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

387  1      submit: procedure(adr) byte;
388  2        declare adr address;
389  2        declare fn based adr (12) byte;
390  2        declare (i,match) byte;

391  2        compare: procedure(j);
392  3          dcl j byte;
393  3          if (fn(j) and 07fh) = sub$fcb(j) then
394  3            return;
395  3          match = false;
396  3        end compare;

397  2        match = true;
398  2          do i = 1 to 3;  /* sub = SYS     $$$ */
399  3          call compare(i);
400  3          call compare(i+8);
401  3          end;
402  2        return match;
403  2      end submit;

         /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

404  1      kill$rsx: procedure;
405  2        declare (fcb$adr,a) address;

406  2        if delimiter <> 9 then    /* check for eoln */
407  2          call error(1);
           /* remove SUBMIT & GET rsx modules */
408  2        do while (fcb$adr:=rsx$call(.get$fcb$pb)) <> 0ffh;
409  3          a = rsx$call(.get$kill$pb);
410  3          if submit(fcb$adr) then
411  3            call print$buf(.('SUBMIT of $'));
             else
412  3            call print$buf(.('GET  from $'));
413  3          call print$fn(fcb$adr);
414  3          call print$buf(.(' stopped$'));
415  3          call crlf;
416  3        end;
417  2        call print$buf(.get$msg);
418  2        call print$buf(.('console$'));
419  2        call mon1(0,0);
420  2      end kill$rsx;

         /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

421  1      end$rsx: procedure;
422  2        declare (a,fcb$adr) address;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
423  2       if delimiter <> 9 then    /* check for eoln */
424  2         call error(1);
425  2       if (fcb$adr := rsx$call(.get$fcb$pb)) <> 0ffh then
426  2         if not submit(fcb$adr) then do;
428  3           a = rsx$call(.get$kill$pb);
429  3           call print$buf(.('GET  from $'));
430  3           call print$fn(fcb$adr);
431  3           call print$buf(.(' stopped$'));
432  3           call crlf;
433  3         end;

            /* determine where console input comes from now */
434  2       call print$buf(.get$msg);
435  2       fcb$adr = rsx$call(.get$fcb$pb);
436  2       if fcb$adr = 0ffh then
437  2         call print$buf(.('console$'));
438  2       else do;
439  3         if getscbbyte(26) < 48 then
440  3           call crlf;                /* console width */
441  3         call print$fn(fcb$adr);
442  3       end;
443  2       call mon1(0,0);
444  2     end end$rsx;

          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

445  1     set$rsx$mode: procedure (bit$value);
446  2       declare bit$value byte;
447  2       declare temp address;
448  2       temp = get$console$mode;
449  2       temp = temp and 111111$00$11111111b;  /* mask off bits to be set */
450  2       if bit$value <> 0 then
451  2         temp = temp or (255 + bit$value);
452  2       call set$console$mode(temp);
453  2     end set$rsx$mode;

          /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

454  1     process$file: procedure(buf$adr);
455  2       declare negate byte;
456  2       declare status address;
457  2       declare buf$adr address;
458  2       declare char based status byte;
459  2       parse$fn.buff$adr = buf$adr;
460  2       parse$fn.fcb$adr = .fcb;
461  2       status = parse(.parse$fn);
462  2       if status = 0ffffh then
463  2         call error(2);  /* bad file */
464  2       if status = 0 then    /* eoln */
465  2         call try$open;  /* try$open does not return */
            else
466  2         buf$ptr = status + 1;   /* position buf$ptr past '[' */
467  2       if char <> '[' then       /* PROCESS OPTIONS */
468  2         call error(4);
469  2       do while ((delimiter<>2) and (delimiter<>9));
470  3         call opt$scanner(.options(0),.options$offset(0),.index);
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
471  3          if index = 4 then do;   /* STATUS */
473  4            if delimiter <> 3 then    /* '=' */
474  4              call error(4);
475  4            call opt$scanner(.options(0),.options$offset(0),.index);
476  4            if index = 5 then     /* CONDITIONAL */
477  4              call set$rsx$mode(0);
478  4            else if index = 6 then    /* FALSE */
479  4              call set$rsx$mode(1);
480  4            else if index = 7 then    /* TRUE */
481  4              call set$rsx$mode(2);
                  else
482  4              call error(0);      /* Not a valid option */
483  4          end;
484  3          else do;    /* ECHO, FILTER, & SYSTEM  options */
485  4            negate=false;
486  4            if index = 15 then do;
488  5              negate = true;
489  5              call opt$scanner(.options(0),.options$offset(0),.index);
490  5            end;
491  4            if index = 16 then do;           /* ECHO */
493  5              if negate then
494  5                getpb.echo$flag = false;
                  else
495  5                getpb.echo$flag = true;
496  5            end;
497  4            else if index = 17 then do;      /* FILTER */
499  5              if negate then
500  5                getpb.filtered$flag = false;
                  else
501  5                getpb.filtered$flag = true;
502  5            end;
503  4            else if index = 18 then do;      /* SYSTEM */
505  5              if negate then
506  5                getpb.program$flag = true;
                  else
507  5                getpb.program$flag = false;
508  5            end;
509  4            else if index = 19 then do;      /* PROGRAM */
511  5              if negate then
512  5                getpb.program$flag = false;
                  else
513  5                getpb.program$flag = true;
514  5            end;
                  else
515  4              call error(0);
516  4          end;
517  3        end;
518  2        call try$open;    /* all set up, so do open */
519  2      end process$file;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

520  1      input$found: procedure (buffer$adr) byte;
521  2        declare buffer$adr address;
522  2        declare char based buffer$adr byte;
523  2        do while (char = ' ') or (char = 9); /* tabs & spaces */
524  3          buffer$adr = buffer$adr + 1;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
525  3       end;
526  2       if char = 0 then  /* eoln */
527  2         return false;   /* input not found */
               else
528  2         return true;    /* input found */
529  2     end input$found;

           /* - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - */

           /*********************************
           *                               *
           *   M A I N   P R O G R A M     *
           *                               *
           *********************************/

530  1     plm:
           do;
531  2       if (low(version) < cpmversion) or (high(version)=1) then do;
533  3         call print$buf(.('Requires CP/M 3.0$'));
534  3         call mon1(0,0);
535  3       end;
536  2       if not input$found(.tbuff(1)) then do;  /* just GET */
538  3         call print$buf(.('CP/M 3 GET Version 3.0',cr,lf,'$'));
539  3         call print$buf(.('Get console input from a file',cr,lf,'$'));
540  3         call print$buf(.('Enter file: $'));
541  3         no$chars = read$console$buf(.tbuff(0),128);
542  3         call crlf;
543  3         tbuff(1) = ' ';       /* blank out nc field */
544  3         tbuff(no$chars+2) = 0;    /* mark eoln */
545  3         if not input$found(.tbuff(1)) then    /* quit, no file name */
546  3           call mon1(0,0);
547  3         do i=1 to no$chars;   /* make input capitals */
548  4           tbuff(i+1) = ucase(tbuff(i+1));
549  4         end;
550  3         begin$buffer = .tbuff(2);
551  3         buf$length = no$chars;
552  3         buf$ptr = .tbuff(2);
553  3         call process$file(.tbuff(2));
554  3       end;
555  2       else do;    /* Get with input */
556  3         i = 1;            /* skip over leading spaces */
557  3         do while (tbuff(i) = ' ');
558  4           i = i + 1;
559  4         end;
560  3         begin$buffer = .tbuff(1); /* note beginning of input */
561  3         buf$length = tbuff(0);    /* note length of input */
562  3         buf$ptr = .tbuff(i);  /* set up for scanner */
563  3         index = 0;
564  3         delimiter = 1;
565  3         call opt$scanner(.options(0),.options$offset(0),.index);
566  3         if (index=10) or (index=11) or (index=14) then do;    /* AUX */
568  4           call opt$scanner(.options(0),.options$offset(0),.index);
569  4           if index = 1 then   /* INPUT */
570  4             call opt$scanner(.options(0),.options$offset(0),.index);
571  4           if index = 2 then   /* FROM */
572  4             call opt$scanner(.options(0),.options$offset(0),.index);
573  4           if index = 3 then do;   /* FILE */
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
575  5              getpb.input$type=aux$type;
576  5              call process$file(buf$ptr);
577  5            end;
578  4            else do;
579  5              if (index=10) or (index=11) or (index=14) then    /* AUX */
580  5                call kill$rsx;
                    else
581  5                call error(3);
582  5            end;
583  4          end;
584  3          else do;  /* not AUX */
585  4            if index = 12 then  /* END  */
586  4              call end$rsx;
587  4            if (index=8) or (index=9) or (index=13) then do;    /* CONSOLE */
589  5              if delimiter = 9 then
590  5                call kill$rsx;
                    else
591  5                call opt$scanner(.options(0),.options$offset(0),.index);
592  5            end;
593  4            if index = 1 then   /* INPUT */
594  4              call opt$scanner(.options(0),.options$offset(0),.index);
595  4            if index = 2 then   /* FROM */
596  4              call opt$scanner(.options(0),.options$offset(0),.index);
597  4            if index = 3 then   /* FILE */
598  4              call process$file(buf$ptr);
599  4            if (index=8) or (index=9) or (index=13) then    /* CONIN:, CONSOLE */
600  4              call kill$rsx;
                  else
601  4              call error(3);
602  4          end;
603  3        end;
604  2      end;
605  1    end get;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____ ____

CROSS-REFERENCE LISTING

DEFN	ADDR	SIZE	NAME, ATTRIBUTES, AND REFERENCES	
352	005CH	2	A	ADDRESS 353 358
405	0066H	2	A	ADDRESS 409
228	0000H	1	A	BYTE BASED(S) 230
422	0068H	2	A	ADDRESS 428
2			ADDR	LITERALLY
387	005FH	2	ADR	ADDRESS PARAMETER 388 389
2			AUXTYPE	LITERALLY 575
6	0003H	2	BEGINBUFFER	ADDRESS 259 550 560
445	006CH	1	BITVALUE	BYTE PARAMETER 446 450 451
2			BKSP	LITERALLY 319
454	006FH	2	BUFADR	ADDRESS PARAMETER 457 459
120	0000H	1	BUFF	BYTE BASED(BUFPTR) ARRAY(1) 137 141 150 153 157
18	0000H	2	BUFFADR	ADDRESS MEMBER(PARSEFN) 378 459
51	0021H	2	BUFFERADDRESS	ADDRESS PARAMETER 52 53 56 57
47	001FH	2	BUFFERADDRESS	ADDRESS PARAMETER 48 49
520	0074H	2	BUFFERADR	ADDRESS PARAMETER 521 522 524
7	0005H	1	BUFLENGTH	BYTE 263 264 551 561

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 14 000BH    2  BUFPTR           ADDRESS
                                 120  161  165  167  214  217  248  268  270  378  466
                                 552  562  576  598

307 0056H    1  C                BYTE
                                 313  314  315  317  319  328  331

302 0054H    1  C                BYTE
                                 303  304

225 0047H    1  C                BYTE PARAMETER
                                 227  229

239 0051H    1  CAROTFLAG        BYTE
                                 250  258  278

  3 0000H    1  CCPFLAG          BYTE

  2             CCPFLAGOFFSET    LITERALLY

294 0053H    1  CHAR             BYTE PARAMETER
                                 295  296  297  298  299

 40 001EH    1  CHAR             BYTE PARAMETER
                                  41   42

522 0000H    1  CHAR             BYTE BASED(BUFFERADR)
                                 523  526

458 0000H    1  CHAR             BYTE BASED(STATUS)
                                 467

165 0000H    1  CHARAC           BYTE BASED(BUFPTR)
                                 166

121 003BH    1  CHARACTER        BYTE
                                 129  137  141  147  150  153  154  157  158  177  183

109 0030H    1  CHARACTER        BYTE PARAMETER
                                 110  114

 63 04D7H    9  CHECKCONSTAT     PROCEDURE BYTE STACK=0002H
                                 331

122 06D9H   86  CHECKINLIST      PROCEDURE STACK=0000H
                                 138

237 0000H    1  CHR1             BYTE BASED(STRINGPTR)
                                 243

238 0000H    1  CHR2             BYTE BASED(TSTRINGPTR)
                                 248

 32 0000H    1  CMDRV            BYTE EXTERNAL(3)

354 005EH    1  CODE             BYTE
                                 367  369
```

CP/M plus Vn 3.0
COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SFR. # CP3-135

```
234 0048H    1  CODE            BYTE PARAMETER
                                 235  265  267  269  283

391 0BD8H   37  COMPARE         PROCEDURE STACK=0000H
                                 399  400

 44 048EH    9  CONIN           PROCEDURE BYTE STACK=0002H
                                 303

  2              CONTYPE        LITERALLY
                                  16

  4 0001H    1  CONWIDTH        BYTE
                                 260  261  262  263  264  275  281

  2              CONWIDTHOFFSET LITERALLY
                                 260

  2              CPMVERSION     LITERALLY
                                 531

  2              CR             LITERALLY
                                 222  315  538  539

221 0805H   11  CRLF            PROCEDURE STACK=0004H
                                 246  256  272  282  291  308  309  372  383  415  432
                                 440  542

  2              CTRLC          LITERALLY

  2              CTRLX          LITERALLY
                                 317

  2              DCL            LITERALLY

 14 0010H    1  DELIMITER       BYTE
                                 154  155  158  166  170  177  179  183  216  218  406
                                 423  469  473  564  589

 14 00BAH   11  DELIMITERS      BYTE ARRAY(11) DATA
                                 112  114

 70 0026H    2  DMA             ADDRESS PARAMETER
                                  71   72

                DOUBLE         BUILTIN
                                  89

336 0000H    1  DRIV            BYTE BASED(FCBAD)
                                 339  341

164 07E9H   28  EATBLANKS       PROCEDURE STACK=0002H
                                 172  215

 16 0001H    1  ECHOFLAG        BYTE MEMBER(GETPB)
                                 494  495
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 14 000EH     1  ENDBUF            BYTE
                                    160  161  188  191  192  214  270

 14 0089H     1  ENDLIST           BYTE DATA
                                    112  126

 15 0011H     1  ENDOFSTRING       BYTE INITIAL
                                    145

421 0C6CH   141  ENDRSX            PROCEDURE STACK=000AH
                                    586

234 083BH   308  ERROR             PROCEDURE STACK=0008H
                                    379  407  424  463  468  474  482  515  581  601

352 005AH     2  ERRORCODE         ADDRESS
                                    355  356  358  360  364  366  367  375

331 0A96H        EXIT              LABEL
                                    316

186 0613H        EXIT1             LABEL
                                    182

225 0046H     1  F                 BYTE PARAMETER
                                    227  230

  2              FALSE             LITERALLY
                                    146  148  194  258  395  485  494  500  507  512  527

 33 0000H     1  FCB               BYTE ARRAY(1) EXTERNAL(4)
                                    364  375  384  460

 34 0000H     1  FCB16             BYTE ARRAY(1) EXTERNAL(5)
                                    311  314  324  362  373

333 0057H     2  FCBAD             ADDRESS PARAMETER
                                    335  336  337

 66 0024H     2  FCBADDRESS        ADDRESS PARAMETER
                                     67   68

422 006AH     2  FCBADR            ADDRESS
                                    425  426  430  435  436  441

 18 0002H     2  FCBADR            ADDRESS MEMBER(PARSEFN)
                                    460

405 0064H     2  FCBADR            ADDRESS
                                    408  410  413

225 0810H    43  FILL              PROCEDURE STACK=0002H
                                    311

 16 0002H     1  FILTEREDFLAG      BYTE MEMBER(GETPB)
                                    500  501
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
213 06A9H        FINISHED          LABEL
                                    198

337 0000H   12   FN                BYTE BASED(FCBAD) ARRAY(12)
                                    347  348

389 0000H   12   FN                BYTE BASED(ADR) ARRAY(12)
                                    393

  2              FOREVER           LITERALLY

121 003DH    1   FOUNDFIRST        BYTE

 20 0000H    1   FUNC              BYTE PARAMETER
                                     21

 24 0000H    1   FUNC              BYTE PARAMETER
                                     25

 28 0000H    1   FUNC              BYTE PARAMETER
                                     29

  1 021FH  607   GET               PROCEDURE STACK=000EH

 92 054AH    9   GETCONSOLEMODE    PROCEDURE ADDRESS STACK=0002H
                                    448

106 0000H        GETF              PROCEDURE EXTERNAL(12) STACK=0000H
                                    385

 11 0009H    1   GETFCBPB          BYTE INITIAL
                                    355  408  425  435

  9 0007H    1   GETINITPB         BYTE INITIAL

 10 0008H    1   GETKILLPB         BYTE INITIAL
                                    360  409  428

 13 000CH   28   GETMSG            BYTE ARRAY(28) DATA
                                    381  417  434

306 0A06H  151   GETPASSWD         PROCEDURE STACK=0006H
                                    371

 16 0012H    4   GETPB             STRUCTURE INITIAL
                                    385  494  495  500  501  506  507  512  513  575

  2              GETRSXFCB         LITERALLY
                                     11

  2              GETRSXINIT        LITERALLY
                                      9

  2              GETRSXKILL        LITERALLY
                                     10

 78 0510H   24   GETSCBBYTE        PROCEDURE BYTE STACK=0002H
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                                260  382  439

301 09F8H   14  GETUCASE        PROCEDURE BYTE STACK=0004H
                                313

191 0623H       GOOD            LABEL
                                185

                HIGH            BUILTIN
                                367  531

390 0061H   1   I               BYTE
                                398  399  400

307 0055H   1   I               BYTE
                                312  314  321  324

  5 0002H   1   I               BYTE
                                547  548  556  557  558  562

235 0049H   1   I               BYTE
                                273

123 0043H   1   I               BYTE
                                124  125  127  128  131

121 0038H   1   I               BYTE
                                140  149  150  153  156  157  160  188  191

119 0036H   2   IDXPTR          ADDRESS PARAMETER
                                120  121

 14 000DH   1   INDEX           BYTE
                                470  471  475  476  478  480  486  489  491  497  503
                                509  563  565  566  568  569  570  571  572  573  579
                                585  587  591  593  594  595  596  597  599

121 0000H   1   INDEX           BYTE BASED(IDXPTR)
                                128  133  175  176  193  195  197  199  210  213

 24 0000H   2   INFO            ADDRESS PARAMETER
                                 26

 28 0000H   2   INFO            ADDRESS PARAMETER
                                 30

 20 0000H   2   INFO            ADDRESS PARAMETER
                                 22

520 0EABH   55  INPUTFOUND      PROCEDURE BYTE STACK=0002H
                                536  545

 12 000AH   1   INPUTTYPE       BYTE

 16 0000H   1   INPUTTYPE       BYTE MEMBER(GETPB)
                                575
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
106 0000H    2  INPUTTYPE        ADDRESS PARAMETER
                                  107

235 004AH    1  J                BYTE
                                  242  247

391 0063H    1  J                BYTE PARAMETER
                                  392  393

121 0039H    1  J                BYTE
                                  203

 14 000FH    1  J                BYTE INITIAL

334 0059H    1  K                BYTE
                                  344  345  347  348

110 0031H    1  K                BYTE
                                  111  112  114  115  116

404 0C00H  108  KILLRSX           PROCEDURE STACK=000AH
                                  580  590  600

 36 0000H    1  LEN0             BYTE EXTERNAL(7)

 38 0000H    1  LEN1             BYTE EXTERNAL(9)

121 0041H    1  LENFOUND         BYTE
                                  192  200  203

121 0040H    1  LENNEW           BYTE
                                  199  200

121 003CH    1  LETTERINWORD     BYTE
                                  139  144  145  147

  2             LF               LITERALLY
                                  223  538  539

121 0000H    1  LIST             BYTE BASED(LISTPTR) ARRAY(1)
                                  126  129  145  147

119 0032H    2  LISTPTR          ADDRESS PARAMETER
                                  120  121

  2             LIT              LITERALLY

112 057CH       LOOP             LABEL
                                  117

                LOW              BUILTIN
                                  366  531

390 0062H    1  MATCH            BYTE
                                  395  397  402

 51 0023H    1  MAX              BYTE PARAMETER
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                                  54  55

     0000H        MEMORY          BYTE ARRAY(0)

  74 0028H     1  MODE            BYTE PARAMETER
                                    75  76

  20 0000H        MON1            PROCEDURE EXTERNAL(0) STACK=0000H
                                    42  49  56  72  76  90  97 292 329 419 443
                                   534 546

  24 0000H        MON2            PROCEDURE BYTE EXTERNAL(1) STACK=0000H
                                    45  64  82

  28 0000H        MON3            PROCEDURE ADDRESS EXTERNAL(2) STACK=0000H
                                    61  68  93 101

 455 0071H     1  NEGATE          BYTE
                                   485 488 493 499 505 511

  53 0000H     2  NEWMAX          ADDRESS BASED(BUFFERADDRESS)
                                    55  58

  95 002CH     2  NEWVALUE        ADDRESS PARAMETER
                                    96  97

 195 0638H        NEXTOPT         LABEL
                                   206 209

 235 004BH     1  NLINES          BYTE
                                   263 273

   8 0006H     1  NOCHARS         BYTE
                                   541 544 547 551

 313 0A2BH        NXTCHR          LABEL
                                   325

 119 0034H     2  OFFPTR          ADDRESS PARAMETER
                                   120 121

  84 002AH     1  OFFSET          BYTE PARAMETER
                                    85  87

  78 0029H     1  OFFSET          BYTE PARAMETER
                                    79  80

  17 0000H     1  OFFSET          BYTE MEMBER(SCBPD)
                                    80  87

 121 0000H     1  OFFSETS         BYTE BASED(OFFPTR) ARRAY(1)
                                   125 131 199

  66 04E0H    16  OPENFILE        PROCEDURE ADDRESS STACK=0002H
                                   364 375

  14 0028H   125  OPTIONS         BYTE ARRAY(125) DATA
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                                 470  475  489  565  568  570  572  591  594  596

  14 00A5H    20  OPTIONSOFFSET  BYTE ARRAY(20) DATA
                                 470  475  489  565  568  570  572  591  594  596

 119 05AAH   303  OPTSCANNER     PROCEDURE STACK=0004H
                                 470  475  489  565  568  570  572  591  594  596

 103 0000H        PARSE          PROCEDURE ADDRESS EXTERNAL(11) STACK=0000H
                                 461

  18 001AH     4  PARSEFN        STRUCTURE
                                 378  459  460  461

  35 0000H     2  PASS0          ADDRESS EXTERNAL(6)

  37 0000H     2  PASS1          ADDRESS EXTERNAL(8)

 103 0000H     2  PFCB           ADDRESS PARAMETER
                                 104

  19 0222H        PLM            LABEL PUBLIC
                                 530

  47 0497H    16  PRINTBUF       PROCEDURE STACK=0002H
                                 279  280  284  285  286  287  288  289  310  338  381
                                 411  412  414  417  418  429  431  434  437  533  538
                                 539  540

  40 047EH    16  PRINTCHAR      PROCEDURE STACK=0002H
                                 222  223  243  251  253  341  342  346  348

 240 096FH   107  PRINTCOMMAND   PROCEDURE STACK=0006H
                                 275  277

 333 0A9DH   104  PRINTFN        PROCEDURE STACK=0004H
                                 384  413  430  441

   2             PROC           LITERALLY
                                 221

 454 0D31H   378  PROCESSFILE    PROCEDURE STACK=000CH
                                 553  576  598

 353 0000H     1  PROGFLAG       BYTE BASED(A)
                                 359

  16 0003H     1  PROGRAMFLAG    BYTE MEMBER(GETPB)
                                 506  507  512  513

  51 04A7H    39  READCONSOLEBUF PROCEDURE BYTE STACK=0002H
                                 541

 235 004CH     1  REM            BYTE
                                 264  277

 311 0A12H        RETRY          LABEL
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
                                318  322

 74 0500H   16  RETURNERRORS      PROCEDURE STACK=0002H
                                 363  365

 99 0563H   16  RSXCALL          PROCEDURE ADDRESS STACK=0002H
                                 355  360  408  409  425  428  435

 99 002EH    2  RSXPB            ADDRESS PARAMETER
                                 100  101

225 0044H    2  S                ADDRESS PARAMETER
                                 226  228  231

121 003FH    1  SAVEINDEX        BYTE
                                 193  213

 17 0016H    4  SCBPD            STRUCTURE
                                  80   81   82   87   88   89   90

109 0573H   55  SEPARATOR        PROCEDURE BYTE STACK=0000H
                                 154  158  166  177  183

 17 0001H    1  SET              BYTE MEMBER(SCBPD)
                                  81   88

 95 0553H   16  SETCONSOLEMODE   PROCEDURE STACK=0002H
                                 452

 70 04F0H   16  SETDMA           PROCEDURE STACK=0002H
                                 362  373

445 0CF9H   56  SETRSXMODE       PROCEDURE STACK=0004H
                                 477  479  481

 84 0528H   34  SETSCBBYTE       PROCEDURE STACK=0002H

136 072FH   30  SETUP            PROCEDURE STACK=0002H
                                 174  186  196

240 0052H    1  SIZE             BYTE PARAMETER
                                 241  242  247

152 078DH   92  SKIP             PROCEDURE STACK=0002H
                                 189  211

 14 00C5H    1  SPACE            BYTE DATA
                                 166

121 003EH    1  START            BYTE
                                 124  173  176  195

456 0072H    2  STATUS           ADDRESS
                                 458  461  462  464  466

236 004DH    2  STRINGPTR        ADDRESS
                                 237  244  259  274
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
 13 0000H   12  SUBFCB          BYTE ARRAY(12) DATA
                                 393

387 0BA9H   50  SUBMIT          PROCEDURE BYTE STACK=0002H
                                 410  426

 39 0000H    1  TBUFF           BYTE ARRAY(1) EXTERNAL(10)
                                 536  541  543  544  545  548  550  552  553  557  560
                                 561  562

447 006DH    2  TEMP            ADDRESS
                                 448  449  451  452

143 074DH   64  TESTLETTER      PROCEDURE STACK=0000H
                                 180  204

  2             TRUE            LITERALLY
                                  16  178  202  250  397  488  495  501  506  513  528

351 0B05H  164  TRYOPEN         PROCEDURE STACK=000AH
                                 465  518

236 004FH    2  TSTRINGPTR      ADDRESS
                                 238  254  259  274

294 09DAH   30  UCASE           PROCEDURE BYTE STACK=0000H
                                 303  548

121 0042H    1  VALID           BYTE
                                 146  148  178  181  194  202  205

 84 002BH    1  VALUE           BYTE PARAMETER
                                  86   89

 17 0002H    2  VALUE           ADDRESS MEMBER(SCBPD)
                                  89

 60 04CEH    9  VERSION         PROCEDURE ADDRESS STACK=0002H
                                 531

121 003AH    1  WRDPOS          BYTE
                                 125  126  129  131  139


MODULE INFORMATION:

    CODE AREA SIZE     = 0EE2H   3810D
    VARIABLE AREA SIZE = 0076H    118D
    MAXIMUM STACK SIZE = 000EH     14D
    939 LINES READ
    0 PROGRAM ERROR(S)

END OF PL/M-80 COMPILATION
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
0000 GET
0000 MCD80A
0004 BDISK     0080 BUFF      0080 BUFFA     0050 CMDRV     0000 CPU
007C CR              006D DOLLA     005C FCB       006C FCB16     005C FCBA
005C IFCB      005C IFCBA     0003 IOBYTE    0053 LEN0      0056 LEN1
0006 MAXB      0006 MEMSIZ    0005 MON1      0005 MON2      0005 MON2A
0005 MON3      0000 OFFSET    006E PARMA     0051 PASS0     0054 PASS1
007F R0              007D RR        007D RRECA     007F RRECO     005C SFCB
0080 TBUFF
0000 GET
1503 MEMORY    148D CCPFLAG   148E CONWIDTH  148F I
1490 BEGINBUFFER             1492 BUFLENGTH 1493 NOCHARS   1494 GETINITPB
1495 GETKILLPB 1496 GETFCBPB  1497 INPUTTYPE 0180 SUBFCB    018C GETMSG
01A8 OPTIONS   0225 OPTIONSOFFSET            0239 ENDLIST   023A DELIMITERS
0245 SPACE     1498 BUFPTR    149A INDEX     149B ENDBUF    149C J
149D DELIMITER 149E ENDOFSTRING              149F GETPB     14A3 SCBPD
14A7 PARSEFN   03A2 FLM       05FE PRINTCHAR 14AB CHAR      060E CONIN
0617 PRINTBUF  14AC BUFFERADDRESS            0627 READCONSOLEBUF
14AE BUFFERADDRESS           14B0 MAX       064E VERSION
0657 CHECKCONSTAT            0660 OPENFILE  14B1 FCBADDRESS 0670 SETDMA
14B3 DMA       0680 RETURNERRORS             14B5 MODE      0690 GETSCBBYTE
14B6 OFFSET    06A8 SETSCBBYTE 14B7 OFFSET   14B8 VALUE
06CA GETCONSOLEMODE          06D3 SETCONSOLEMODE           14B9 NEWVALUE
06E3 RSXCALL   14BB RSXPB     06F3 SEPARATOR 14BD CHARACTER 14BE K
06FC LOOP      072A OPTSCANNER 14BF LISTPTR  14C1 OFFPTR    14C3 IDXPTR
14C5 I         14C6 J         14C7 WRDPOS    14C8 CHARACTER
14C9 LETTERINWORD            14CA FOUNDFIRST 14CB START     14CC SAVEINDEX
14CD LENNEW    14CE LENFOUND  14CF VALID     0859 CHECKINLIST
14D0 I         08AF SETUP     08CD TESTLETTER 090D SKIP      0969 EATBLANKS
0793 EXIT1     07A3 GOOD      07B8 NEXTOPT   0829 FINISHED  0985 CRLF
0990 FILL      14D1 S         14D3 F         14D4 C         09BB ERROR
14D5 CODE      14D6 I         14D7 J         14D8 NLINES    14D9 REM
14DA STRINGPTR 14DC TSTRINGPTR 14DE CAROTFLAG 0AEF PRINTCOMMAND
14DF SIZE      0B5A UCASE     14E0 CHAR      0B78 GETUCASE  14E1 C
0B86 GETPASSWD 14E2 I         14E3 C         0B92 RETRY     0BAB NXTCHR
0C16 EXIT      0C1D PRINTFN   14E4 FCBAD     14E6 K         0C85 TRYOPEN
14E7 ERRORCODE 14E9 A         14EB CODE      0D29 SUBMIT    14EC ADR
14EE I         14EF MATCH     0D5B COMPARE   14F0 J         0D80 KILLRSX
14F1 FCBADR    14F3 A         0DEC ENDRSX    14F5 A         14F7 FCBADR
0E79 SETRSXMODE       14F9 BITVALUE  14FA TEMP      0EB1 PROCESSFILE
14FC BUFADR    14FE NEGATE    14FF STATUS    102B INPUTFOUND 1501 BUFFERADR
0000 PARSE
11AB DBLNK1    119F DEBLNK    115D DELIM     1198 DELIM1    119B DELIM2
1155 PADFLD    10F3 PARS81    10F4 PARS82    1062 PARSE     1083 PARSE1
10B1 PARSE2    10B4 PARSE3    10C6 PARSE4    10CA PARSE5    10CC PARSE6
10E0 PARSE7    10E4 PARSE8    1105 PARSE9    112E PWFLD     1141 PWFLD1
1120 SETFD1    1122 SETFD2    1126 SETFD3    110B SETFLD
0000 GETF
FFFF BIOSFU    007F BIOSON    00B4 CCPFLG    0001 CINF      0010 CLOSEF
00F9 COMMON    0074 CONINF    00CF CONMOD    007A CONOUF    006E CONSTF
0002 COUTF     000D CR        0006 CRAWF     000A CREADF    000B CSTATF
0019 CURDRV    0013 DELF      001A DMAF      0014 DREADF    00AA ERRFLG
0000 EVERYT    0000 FALSE     0080 INITF     008D JKILLF    00C3 JMPI
0081 KILLF     000A LF        00D4 LISTCP    0080 LISTFX    0021 LXIH
00FF NOREST    000F OPENF     0009 PBUFF     0005 PCHRF     0098 REALDO
003C RSXF      003A SCBA      0031 SCBF      0022 SHLDI     0080 STFIX
FFFF TRUE      0020 USERF     0000 WBOOT     0001 WBOOTA    0068 WBOOTF
12C1 BIOSER    138E BIOSIN    1390 BIOSMO    138C BIOSTA    1396 CONINJ
1392 CONSTJ    139D COOKED    1303 DATA49    139C ECHO      12D5 ERROR
12DA ERROR1    1253 FIX0      1260 FIX1      1273 FIX2      127D FIX3
11AF GETF      12FB GETSCB    12F3 GETUSR    138A INITTA    1310 JOURNK
1305 LDIR      136D MEMERR    13C1 MOVEND    11E8 MOVFCB    138A MOVSTA
1312 NOBIOS    12ED NOMEM     1337 NOTDON    12D2 NOTIMP    139E PROGRA
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

1399 REAL3D	1391 RESTOR	1311 RSXINI	12AB RSXMOV	138A SCBADR
12F5 SETUSR	13C0 SUBCR	13A0 SUBFCB	13AE SUBMOD	13AF SUBRC
139F SUBUSR	127F TRAP			

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____ ____

```
0000 GET#
0000 MCDBOA#
0000 GET#
05FE 40         0602 42         060D 43         060E 44         060E 45
0617 46         0617 47         061D 49         0626 50         0627 51
062F 55         0639 56         0642 57         0649 58         064E 59
064E 60         064E 61         0657 62         0657 63         0657 64
0660 65         0660 66         0666 68         0670 69         0670 70
0676 72         067F 73         0680 74         0684 76         068F 77
0690 78         0694 80         069A 81         069F 82         06A8 83
06A8 84         06AE 87         06B4 88         06B9 89         06C1 90
06C9 91         06CA 92         06CA 93         06D3 94         06D3 95
06D9 97         06E2 98         06E3 99         06E9 101        06F3 102
06F3 109        06F7 111        06FC 112        070B 113        070E 114
071E 115        0722 116        0726 117        0729 118        072A 119
0859 122        0859 124        085F 125        086A 126        087B 127
087F 128        0886 129        0897 130        0898 131        08A6 132
08A9 133        08AE 134        08AF 135        08AF 136        08AF 137
08B6 138        08B9 139        08BF 140        08C4 141        08CC 142
08CD 143        08CD 144        08D1 145        08DF 146        08E7 147
08F8 148        08FD 149        0901 150        090C 151        090D 152
090D 153        091B 154        0925 155        093A 156        093E 157
0949 158        0953 159        0956 160        095C 161        0968 162
0969 163        0969 164        0969 166        097A 167        0981 168
0984 169        073B 170        0743 171        0744 172        0747 173
074C 174        074F 175        0758 176        075F 177        0769 178
076E 179        0776 180        0779 181        0780 182        0783 183
078D 184        0790 185        0793 186        0796 187        0799 188
079F 189        07A2 190        07A3 191        07A9 192        07AC 193
07B3 194        07B8 195        07BF 196        07C2 197        07CB 198
07CE 199        07EA 200        07F2 202        07F7 203        0806 204
0809 205        0810 206        0813 207        081A 208        081D 209
0820 210        0825 211        0828 212        0829 213        0830 214
083E 215        0841 216        0849 217        0853 218        0858 219
0859 220        0985 221        0985 222        098A 223        098F 224
0990 225        099D 229        09A9 230        09B0 231        09B7 232
09BA 233        09BB 234        0AEF 240        0AF3 242        0B02 243
0B09 244        0B10 245        0B17 246        0B1A 247        0B29 248
0B36 250        0B3B 251        0B40 252        0B43 253        0B48 254
0B4F 255        0B56 256        0B59 257        09BF 258        09C4 259
09CD 260        09D5 261        09DD 262        09E2 263        09F4 264
0A02 265        0A0A 267        0A22 268        0A2C 269        0A34 270
0A43 271        0A43 272        0A46 273        0A55 274        0A5B 275
0A62 276        0A69 277        0A70 278        0A77 279        0A80 280
0A86 281        0A8E 282        0A91 283        0AA1 284        0AAA 285
0AB3 286        0ABC 287        0AC5 288        0ACE 289        0AD7 290
0AE3 291        0AE6 292        0AEE 293        0B5A 294        0B5E 296
0B66 297        0B6E 298        0B74 299        0B78 300        0B78 301
0B78 303        0B82 304        0B86 305        0B86 306        0B86 308
0B89 309        0B8C 310        0B92 311        0B9D 312        0BAB 313
0BB6 314        0BC3 315        0BCB 31?        0BCE 317        0BD6 318
0BD9 319        0BE1 321        0BE9 322        0BEC 324        0BFC 325
0BFF 326        0BFF 327        0BFF 328        0C07 329        0C0F 330
0C16 331        0C1C 332        0C1D 333        0C23 338        0C29 339
0C32 341        0C3C 342        0C41 343        0C41 344        0C4F 345
0C57 346        0C5C 347        0C6C 348        0C7D 349        0C84 350
0C85 351        0C85 355        0C8E 356        0C9A 358        0CA2 359
0CA7 360        0CB0 361        0CB0 362        0CB6 363        0CBB 364
0CC4 365        0CC9 366        0CD2 367        0CDE 369        0CE6 371
0CE9 372        0CEC 373        0CF2 374        0CF2 375        0CFB 376
0CFE 378        0D04 379        0D09 380        0D09 381        0D0F 382
0D19 383        0D1C 384        0D22 385        0D28 386        0D29 387
0D5B 391        0D5F 393        0D79 394        0D7A 395        0D7F 396
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
0D2F 397    0D34 398    0D40 399    0D47 400    0D50 401
0D57 402    0D5B 403    0D80 404    0D80 406    0D88 407
0D8D 408    0D9F 409    0DA8 410    0DB4 411    0DBD 412
0DC3 413    0DCB 414    0DD1 415    0DD4 416    0DD7 417
0DDD 418    0DE3 419    0DEB 420    0DEC 421    0DEC 423
0DF4 424    0DF9 425    0E0B 426    0E17 428    0E20 429
0E26 430    0E2E 431    0E34 432    0E37 433    0E37 434
0E3D 435    0E46 436    0E52 437    0E5B 439    0E65 440
0E68 441    0E70 442    0E70 443    0E78 444    0E79 445
0E7D 448    0E83 449    0E91 450    0E99 451    0EA8 452
0EB0 453    0EB1 454    0EB7 459    0EBD 460    0EC3 461
0ECC 462    0ED9 463    0EDE 464    0EEA 465    0EF0 466
0EF7 467    0F00 468    0F05 469    0F1D 470    0F2A 471
0F32 473    0F3A 474    0F3F 475    0F4C 476    0F54 477
0F5C 478    0F64 479    0F6C 480    0F74 481    0F7C 482
0F81 483    0F84 485    0F89 486    0F91 488    0F96 489
0FA3 490    0FA3 491    0FAB 493    0FB2 494    0FBA 495
0FBF 496    0FC2 497    0FCA 499    0FD1 500    0FD9 501
0FDE 502    0FE1 503    0FE9 505    0FF0 506    0FF8 507
0FFD 508    1000 509    1008 511    100F 512    1017 513
101C 514    101F 515    1024 516    1024 517    1027 518
102A 519    102B 520    1031 523    1048 524    104F 525
1052 526    105B 527    105E 528    1061 529    039F 530
03A5 531    03BD 533    03C3 534    03CB 535    03CB 536
03D5 538    03DB 539    03E1 540    03E7 541    03F2 542
03F5 543    03FA 544    0405 545    040F 546    0417 547
0426 548    043D 549    0444 550    044A 551    0450 552
0453 553    0459 554    045C 556    0461 557    0470 558
0474 559    0477 560    047D 561    0482 562    048D 563
0492 564    0497 565    04A4 566    04C8 568    04D5 569
04DD 570    04EA 571    04F2 572    04FF 573    0507 575
050C 576    0514 577    0517 579    053B 580    0541 581
0546 582    0546 583    0549 585    0551 586    0554 587
0578 589    0580 590    0586 591    0593 592    0593 593
059B 594    05A8 595    05B0 596    05BD 597    05C5 598
05CD 599    05F1 600    05F7 601    05FC 602    05FC 603
05FC 605
0000 PARSE#

0000 GETF#
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                title 'GET.RSX 3.0 - CP/M 3.0 Input Redirection - August 1982'
                ;*****************************************************************
                ;
                ;       get  'Input Redirection Facility'  version 3.0
                ;
                ;       11/30/82 - Doug Huskey
                ;       This RSX redirects console input and status from a file.
                ;*****************************************************************
                ;
                ;
FFFF =          true            equ     0ffffh
0000 =          false           equ     00000h
                ;
0000 =          submit          equ     false   ;true if submit RSX
0000 =          remove$rsx      equ     false   ;true if RSX removes itself
                ;                               ;false if LOADER does removes
                ;
                ;
                ;       generation procedure
                ;
                ;       rmac getrsx
                ;       xref getrsx
                ;       link getrsx[op]
                ;       ERA get.RSX
                ;       REN get.RSX=getRSX.PRL
                ;       GENCOM $1.COM get.RSX           ($1 is either SUBMIT or GET)
                ;
                ;
                ;       initialization procedure
                ;
                ;       GETF makes a RSX function 60 call with a sub-function of
                ;       128.  GETRSX returns the address of a data table containing:
                ;
                ;       init$table:
                ;               dw      kill            ;RSX remove flag addr in GET
                ;               dw      bios$constat    ;bios entry point in GET
                ;               dw      bios$conin      ;bios entry point in GET
                ;
                ;       GETF initializes the data are between movstart: and movend:
                ;       and moves it into GET.RSX.  This means that data should not
                ;       be reordered without also changing GETF.ASM.
                ;
FFFF =          bios$functions equ      true    ;intercept BIOS console functions
                ;
                ;       low memory locations
                ;
0000 =          wboot   equ     0000h
0005 =          bdos    equ     0005h
0006 =          bdosl   equ     bdos+1
0080 =          buf     equ     0080h
                ;
                ;       equates for non graphic characters
                ;
0003 =          ctlc    equ     03h     ; control c
0005 =          ctle    equ     05h     ; physical eol
0008 =          ctlh    equ     08h     ; backspace
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
 57  0010 =     ctlp   equ    10h     ; prnt toggle
 58  0012 =     ctlr   equ    12h     ; repeat line
 59  0013 =     ctls   equ    13h     ; stop/start screen
 60  0015 =     ctlu   equ    15h     ; line delete
 61  0018 =     ctlx   equ    18h     ; =ctl-u
 62                    if submit
 63             ctlz   equ    0ffh
 64                    else
 65  001A =     ctlz   equ    1ah     ; end of file
 66                    endif
 67  007F =     rubout equ    7fh     ; char delete
 68  0009 =     tab    equ    09h     ; tab char
 69  000D =     cr     equ    0dh     ; carriage return
 70  000A =     lf     equ    0ah     ; line feed
 71  005E =     ctl    equ    5eh     ; up arrow
 72             ;
 73             ;      BDOS function equates
 74             ;
 75  0001 =     cinf   equ    1       ;read character
 76  0002 =     coutf  equ    2       ;output character
 77  0006 =     crawf  equ    6       ;raw console I/O
 78  000A =     creadf equ    10      ;read buffer
 79  000B =     cstatf equ    11      ;status
 80  0005 =     pchrf  equ    5       ;print character
 81  0009 =     pbuff  equ    9       ;print buffer
 82  000F =     openf  equ    15      ;open file
 83  0010 =     closef equ    16      ;close file
 84  0013 =     delf   equ    19      ;delete file
 85  0014 =     dreadf equ    20      ;disk read
 86  001A =     dmaf   equ    26      ;set dma function
 87  0020 =     userf  equ    32      ;set/get user number
 88  0031 =     scbf   equ    49      ;set/get system control block word
 89  003B =     loadf  equ    59      ;loader function call
 90  003C =     rsxf   equ    60      ;RSX function call
 91  0080 =     ginitf equ    128     ;GET initialization sub-function no.
 92  0081 =     gkillf equ    129     ;GET delete sub-function no.
 93  0082 =     gfcbf  equ    130     ;GET file display sub-function no.
 94  0084 =     pinitf equ    132     ;PUT initialization sub-funct no.
 95  0085 =     pckillf       equ    133     ;PUT CON: delete sub-function no.
 96  0086 =     pcfcbf equ    134     ;return PUT CON: fcb address
 97  0089 =     plkillf       equ    137     ;PUT LST: delete sub-function no.
 98  008A =     plfcbf equ    138     ;return PUT LST:fcb address
 99  008C =     gsigf  equ    140     ;signal GET without [SYSTEM] option
100  008D =     jinitf equ    141     ;JOURNAL initialization sub-funct no.
101  008E =     jkillf equ    142     ;JOURNAL delete sub-function no.
102  008F =     jfcbf  equ    143     ;return JOURNAL fcb address
103             ;
104             ;      System Control Block definitions
105             ;
106  003A =     scba   equ    03ah    ;offset of scbadr from SCB base
107  00B3 =     ccpflg equ    0b3h    ;offset of ccpflags word from page boundary
108  0020 =     ccpres equ    020h    ;ccp resident flag = bit 5
109  00FE =     bdosoff equ   0feh    ;offset of BDOS address from page boundary
110  00AC =     errflg equ    0ach    ;offset of error flag from page boundary
111  00C8 =     pg$mode       equ    0c8h    ;offset of page mode byte from pag. bound.
112  00C9 =     pg$def equ    0c9h    ;offset of page mode default from pag. bound.
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
00CF =          conmode         equ     0cfh    ;offset of console mode word from pag. bound.
00D4 =          listcp equ      0d4h    ;offset of ^P flag from page boundary
00D8 =          dmaad  equ      0d8h    ;offset of DMA address from pg bnd.
00E0 =          usrcode         equ     0e0h    ;offset of user number from pg bnd.
00E1 =          dcnt   equ      0e1h    ;offset of dcnt, searcha & searchl from pg bnd.
006E =          constfx         equ     06eh    ;offset of constat JMP from page boundary
0074 =          coninfx         equ     074h    ;offset of conin JMP from page boundary


                ;*****************************************************************
                ;               RSX HEADER
                ;*****************************************************************

0000 000000000serial:           db      0,0,0,0,0,0

                trapjmp:
0006 C34600             jmp     trap            ;trap read buff and DMA functions
0009 C30000     next:   jmp     0               ;go to BDOS
000C 0500       prev:   dw      bdos
000E FF         kill:   db      0FFh            ;0FFh => remove RSX at wstart
000F 00         nbank:  db      0
0010 4745542020rname:   db      'GET     '      ;RSX name
0018 0000       space:  dw      0
001A 00         patch:  db      0

                ;*****************************************************************
                ;               START OF CODE
                ;*****************************************************************

                ;
                ;       ABORT ROUTINE
                ;
                getout:
                        ;
                if bios$functions
                        ;
                        ;restore bios jumps
001B 3ACC02             lda     restore$mode            ;may be FF, 7f, 80 or 0
001E 3C                 inr     a
001F C8                 rz                              ; FF = no bios interception
0020 2ACA02             lhld    biosin
0023 EB                 xchg
0024 2AC802             lhld    biosta
0027 CDCD02             call    restore$bios            ;restore BIOS constat & conin jmps
002A F8                 rm                              ; 7f = RESBDOS jmps not changed
002B 2AC602             lhld    scbadr
002E 2E6E               mvi     l,constfx
0030 36C3               mvi     m,jmp
0032 E8                 rpe                             ; 80 = conin jmp not changed
0033 2E74               mvi     l,coninfx
0035 36C3               mvi     m,jmp
                endif
0037 C9                 ret                             ; 0  = everything done
                ;
                ;       ARRIVE HERE ON EACH BIOS CONIN OR CONSTAT CALL
                ;
```

COPYRIGHT © 198
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        ;
                        bios$constat:
                                ;
                        if bios$functions
                                ;
                                ;enter here from BIOS constat
0038 010B04             lxi     b,4*256+cstatf  ;b=offset in exit table
003B C34C00             jmp     bios$trap
                        endif
                        ;
                        bios$conin:
                                ;
                        if bios$functions
                                ;
                                ;enter here from BIOS conin
003E 010606             lxi     b,6*256+crawf   ;b=offset in exit table
0041 1EFD               mvi     e,0fdh
0043 C34C00             jmp     biostrap
                        endif
                        ;
                        ;       ARRIVE HERE AT EACH BDOS CALL
                        ;
                        trap:
                                ;
                                ;
0046 219003             lxi     h,excess
0049 0600               mvi     b,0
004B 70                 mov     m,b
                        biostrap:
                                ;enter here on BIOS calls
                
004C E1                 pop     h               ;return address
004D E5                 push    h               ;back to stack
004E 3A0800             lda     trap$jmp+2      ;GET.RSX page address
0051 BC                 cmp     h               ;high byte of return address
0052 DA7E00             jc      exit            ;skip calls on bdos above here
0055 79                 mov     a,c             ;function number
                                ;
                                ;
0056 FE0B               cpi     cstatf          ;status
0058 CAD300             jz      intercept
005B FE06               cpi     crawf
005D CAD300             jz      intercept       ;raw I/O
0060 21C302             lxi     h,statflg       ;zero conditional status flag
0063 3600               mvi     m,0
0065 FE01               cpi     cinf
0067 CAD300             jz      intercept       ;read character
006A FE0A               cpi     creadf
006C CAD300             jz      intercept       ;read buffer
006F FE3C               cpi     rsxf
0071 CA8A00             jz      rsxfunc         ;rsx function
0074 FE1A               cpi     dmaf
0076 C27E00             jnz     exit            ;skip if not setting DMA
0079 EB                 xchg
007A 227F03             shld    udma            ;save user's DMA address
007D EB                 xchg
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        ;
                        exit:
                              ;go to real BDOS

                        if not bios$functions
                              ;
                              jmp     next            ;go to next RSX or BDOS

                        else
007E 78                       mov     a,b             ;get type of call:
007F 21C402                   lxi     h,exit$table    ;0=BDOS call, 4=BIOS CONIN, 6=BIOS CONSTAT
0082 CD1602                   call    addhla
0085 46                       mov     b,m             ;low byte to b
0086 23                       inx     h
0087 66                       mov     h,m             ;high byte to h
0088 68                       mov     l,b             ;HL = .exit routine
0089 E9                       pchl                    ;gone to BDOS or BIOS
                        endif
                        ;
                        ;
                        rsxfunc:                      ;check for initialize or delete RSX functions
008A 1A                       ldax    d               ;get RSX sub-function number
008B 21C602                   lxi     h,init$table    ;address of area initialized by COM file
008E FE80                     cpi     ginitf
0090 C8                       rz
0091 3A0E00                   lda     kill
0094 B7                       ora     a
0095 C27E00                   jnz     exit
0098 1A                       ldax    d
0099 FE82                     cpi     gfcbf
009B 21DC02                   lxi     h,subfcb
009E C8                       rz
                        cksig:
009F FE8C                     cpi     gsigf
00A1 C2AB00                   jnz     ckKill
00A4 218103                   lxi     h,get$active
00A7 3E81                     mvi     a,gKillf
00A9 96                       sub     m               ;toggle get$active flag
00AA 77                       mov     m,a             ;gKillf->0    0->gKillf

                        ckKill:
00AB FE81                     cpi     gKillf          ;remove this instance of GET?
00AD C27E00                   jnz     exit            ;jump if not


                        restor:
00B0 3A8103                   lda     get$active
00B3 B7                       ora     a
00B4 C8                       rz
00B5 CD1B00                   call    getout          ;bios jump fixup

                        if submit
                              mvi     c,closef
                              call    subdos
                              mvi     c,delf
                              call    subdos          ;delete SYSIN??.$$$ if not
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                        endif
00B8 210E00                     lxi     h,kill
00BB 35                         dcr     m               ;set to 0ffh, so we are removed
00BC EB                         xchg                    ; D = base of this RSX
00BD 2AC602                     lhld    scbadr
00C0 2EB4                       mvi     l,ccpflg+1      ;hl = .ccp flag 2 in SCB
00C2 7E                         mov     a,m
00C3 E6BF                       ani     0bfh
00C5 77                         mov     m,a             ;turn off redirection flag
                                ;we must remove this RSX if it is the lowest one
00C6 3A0700                     lda     bdosl+1         ;location 6 high byte
00C9 BA                         cmp     d               ;Does location 6 point to us
00CA C0                         RNZ                     ;return if not
                        if remove$rsx
                                xchg                    ;D = scb page
                                lhld    next+1
                                shld    bdosl
                                xchg                    ;H = scb page
                                mvi     l,bdosoff       ;HL = 'BDOS' address in SCB
                                mov     m,e             ;put next address into SCB
                                inx     h
                                mov     m,d
                                xchg
                                mvi     l,0ch           ;HL = .previous RSX field in next RSX
                                mvi     m,7
                                inx     h
                                mvi     m,0             ;put previous into previous
                                ret
                        else
                                ;       CP/M 3 loader does RSX removal if DE=0
00CB 0E3B                       mvi     c,loadf
00CD 110000                     lxi     d,0
00D0 C30900                     jmp     next            ;ask loader to remove me
                        endif

                        ;
                        ;
                        ;       INTERCEPT EACH BDOS CONSOLE INPUT FUNCTION CALL HERE
                        ;
                        ;       enter with funct in A, info in DE
                        ;
                        intercept:
                        ;
00D3 3A0E00                     lda     kill
00D6 B7                         ora     a
00D7 C27E00                     jnz     exit            ;skip if remove flag turned on
                                ;
                                ;switch stacks
00DA 210000                     lxi     h,0
00DD 39                         dad     sp
00DE 229103                     shld    old$stack
00E1 31F903                     lxi     sp,stack
00E4 C5                         push    b               ;save function #
00E5 D5                         push    d               ;save info
                                ;check redirection mode
00E6 CDFC01                     call    getmode         ;returns with H=SCB page
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
00E9 FE02          cpi     2
00EB CA5701        jz      skip            ;skip if no redirection flag on

                   if submit
                   ;
                   ;       SUBMIT PROCESSOR
                   ;
                           ;check if CCP is calling
                   ckccp:  mvi     l,pg$mode
                           mov     m,H             ;set to non-zero for no paging
                           mvi     l,ccpflg+1      ;CCP FLAG 2 in SCB
                           mov     a,m             ;ccp flag byte 2 to A
                           ori     040h
                           mov     m,a             ;set redirection flag on
                           ani     ccpres          ;zero flag set if not CCP calling
                           lda     ccp$line
                           jz      not$ccp
                           ;yes, CCP is calling
                           ora     a
                           jnz     redirect        ;we have a CCP line
                           ;CCP & not a CCP line
                           push    h
                           call    coninf          ;throw away until next CCP line
                           lxi     h,excess
                           mov     a,m
                           ora     a               ;is this the first time?
                           mvi     m,true
                           lxi     d,garbage
                           mvi     c,pbuff
                           cz      next            ;print the warning if so
                           pop     h
                           lda     Kill
                           ora     a
                           jz      ckccp           ;get next character (unless eof)
                           mov     a,m
                           ani     7fh             ;turn off disk reset (CCP) flag
                           mov     m,a
                           jmp     wboot           ;skip if remove flag turned on
                   ;
                   not$ccp:
                           ;no, its not the CCP
                           ora     a
                           jnz     skip            ;skip if no program line

                   else
00EE 3ADA02                lda     program
00F1 B7                    ora     a               ;program input only?
00F2 2EB4                  mvi     l,ccpflg+1      ;CCP FLAG 2 in SCB
00F4 7E                    mov     a,m             ;ccp flag byte 2 to A
00F5 CA0F01                jz      set$no$page     ;jump if [system] option
                           ;check if CCP is calling
00F8 E620                  ani     ccpres          ;zero flag set if not CCP calling
00FA CA1501                jz      redirect        ;jump if not the CCP
00FD 219303                lxi     h,ccpcnt        ;decrement once for each
0100 35                    dcr     m               ;time CCP active
0101 FCB000                cm      restor          ;if 2nd CCP appearance
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 393  0104 11A000          lxi     d,cksig+1
 394  0107 0E3C            mvi     c,rsxf          ;terminate any GETs waiting for
 395  0109 CD0900          call    next            ;us to finish
 396  010C C35701          jmp     skip
 397                       ;
 398                  set$no$page:
 399  010F F640            ori     40h             ;A=ccpflag2, HL=.ccpflag2
 400  0111 77              mov     m,a             ;set redirection flag on
 401  0112 2EC8            mvi     l,pg$mode
 402  0114 74              mov     m,h             ;set to non-zero for no paging
 403                  endif
 404                       ;
 405                       ;       REDIRECTION PROCESSOR
 406                       ;
 407                  redirect:
 408                       ;break if control-C typed on console
 409  0115 CD4501          call    break
 410  0118 D1              pop     d
 411  0119 C1              pop     b               ;recover function no. & info
 412  011A C5              push    b               ;save function
 413  011B D5              push    d               ;save info
 414  011C 79              mov     a,c             ;function no. to A
 415  011D 216801          lxi     h,retmon        ;program return routine
 416  0120 E5              push    h               ;push on stack
 417                       ;
 418                       ;
 419  0121 FE0A            cpi     creadf
 420  0123 CA6702          jz      func10          ;read buffer (returns to retmon)
 421  0126 FE01            cpi     cinf
 422  0128 CA1B02          jz      func1           ;read character (returns to retmon)
 423  012B FE0B            cpi     cstatf
 424  012D CA3D02          jz      func11          ;status (returns to retmon)
 425                  ;
 426                  func6:
 427                       ;direct console i/o - read if 0ffh
 428                       ;returns to retmon
 429  0130 7B              mov     a,e
 430  0131 3C              inr     a
 431  0132 CA7301          jz      dirinp          ;0ffh in E for status/input
 432  0135 3C              inr     a
 433  0136 CA3D02          jz      CONBRK          ;0feh in E for status
 434  0139 21C302          lxi     h,statflg
 435  013C 3600            mvi     m,0
 436  013E 3C              inr     a
 437  013F CA7801          jz      coninf          ;0fdh in E for input
 438                       ;
 439                       ;direct output function
 440                       ;
 441  0142 C36101          jmp     skip1
 442                       ;
 443                  break: ;
 444                       ;quit if ^C typed
 445  0145 0E0B            mvi     c,cstatf
 446  0147 CDD502          call    real$bdos
 447  014A B7              ora     a               ;was ^C typed?
 448  014B C8              rz
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 449  014C E1              pop     h               ;throw away return address
 450  014D CDB000          call    restor          ;remove this RSX, if so
 451  0150 0E06            mvi     c,crawf
 452  0152 1EFF            mvi     e,0ffh
 453  0154 CD0900          call    next            ;eat ^C if not nested
 454                       ;
 455                  skip:  ;
 456                       ;reset ^C status mode
 457  0157 CDFC01          call    getmode         ;returns .conmode+1
 458  015A 2B              dcx     h               ;hl = .conmode in SCB
 459  015B 7E              mov     a,m
 460  015C E6FE            ani     0feh            ;turn off control C status
 461  015E 77              mov     m,a
 462                       ;restore the BDOS call
 463  015F D1              pop     d               ;restore BDOS function no.
 464  0160 C1              pop     b               ;restore BDOS parameter
 465                       ;restore the user's stack
 466  0161 2A9103     skip1: lhld    old$stack
 467  0164 F9              sphl
 468  0165 C37E00          jmp     exit            ;goto BDOS
 469
 470                  ;
 471                  retmon:
 472                       ;normal entry point, char in A
 473  0168 FE1A            cpi     ctlz
 474  016A CA5701          jz      skip
 475  016D 2A9103          lhld    old$stack
 476  0170 F9              sphl
 477  0171 6F              mov     l,a
 478  0172 C9              ret                     ;to calling program
 479
 480
 481                  ;*****************************************************************
 482                  ;               BIOS FUNCTIONS (REDIRECTION ROUTINES)
 483                  ;*****************************************************************
 484                  ;
 485                  ;       ;direct console input
 486                  dirinp:
 487  0173 CD3D02          call    conbrk
 488  0176 B7              ora     a
 489  0177 C8              rz
 490                  ;
 491                  ;
 492                  ;       get next character from file
 493                  ;
 494                       ;
 495                  coninf:
 496                  getc:  ;return ^Z if end of file
 497  0178 AF              xra     a
 498  0179 21FD02          lxi     h,cbufp         ;cbuf index
 499  017C 34              inr     m               ;next chr position
 500  017D FCAE01          cm      readf           ;read a new record
 501  0180 B7              ora     a
 502  0181 061A            mvi     b,ctlz          ;EOF indicator
 503  0183 C29001          jnz     getc1           ;jump if end of file
 504  0186 3AFD02          lda     cbufp
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 505  0189 21FF02          lxi     h,cbuf
 506  018C CD1602          call    addhla          ;HL = .char
 507                       ;one character look ahead
 508                       ;new char in B, current char in nextchr
 509  018F 46              mov     b,m             ;new character in B
 510  0190 78        getc1: mov     a,b
 511  0191 FE1A            cpi     ctlz
 512  0193 C5              push    b
 513  0194 CC8000          cz      restor
 514  0197 C1              pop     b
 515  0198 21FE02          lxi     h,nextchr
 516  019B 7E              mov     a,m             ;current character
 517  019C FE0D            cpi     cr
 518  019E 70              mov     m,b             ;save next character
 519  019F C0              rnz
 520  01A0 78              mov     a,b             ;A=character after CR
 521  01A1 FE0A            cpi     lf              ;is it a line feed
 522  01A3 CC7801          cz      getc            ;eat line feeds after a CR
 523                                               ;this must return from above
 524                                               ;rnz because nextchr = lf
 525                       ;
 526                  if submit
 527                       ;
 528                       mov     a,b             ;get nextchr
 529                       sui     '<'             ;program line?
 530                       sta     ccp$line        ;zero if so
 531                       cz      getc            ;eat '<' char
 532                                               ;this must return from above
 533                                               ;rnz because nextchr = <
 534                  endif
 535  01A6 3E0D            mvi     a,cr            ;get back the cr
 536  01A8 C9              ret                     ;with character in a
 537                  ;
 538                  ;    set DMA address in DE
 539                  ;
 540  01A9 0E1A       setdma:      mvi     c,dmaf
 541  01AB C30900          jmp     next
 542                  ;
 543                  ;    read next record
 544                  ;
 545  01AE 0E14       readf: mvi     c,dreadf        ;read next record of input to cbuf
 546  01B0 C5         subdos:      push    b
 547  01B1 11FF02          lxi     d,cbuf
 548  01B4 CDA901          call    setdma          ;set DMA to our buffer
 549  01B7 2AC602          lhld    scbadr
 550  01BA 118203          lxi     d,sav$area      ;10 byte save area
 551  01BD C1              pop     b               ;C  = function no.
 552  01BE E5              push    h               ;save for restore
 553  01BF D5              push    d               ;save for restore
 554  01C0 CD0B02          call    mov7            ;save hash info in save area
 555  01C3 2EE0            mvi     l,usrcode       ;HL = .dcnt in SCB
 556  01C5 CD0B02          call    mov7            ;save dcnt, searcha & 1, user# &
 557  01C8 2B              dcx     h               ;multi-sector I/O count
 558  01C9 3601            mvi     m,1             ;set multi-sector count = 1
 559  01CB 11DB02          lxi     d,subusr        ;DE = .submit user #
 560  01CE 2EE0            mvi     l,usrcode       ;HL = .BDOS user number
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
 561  01D0 1A            ldax   d
 562  01D1 77            mov    m,a
 563  01D2 13            inx    d
 564  01D3 CD0900        call   next          ;read next record
 565  01D6 E1            pop    h             ;HL = .sav$area
 566  01D7 D1            pop    d             ;DE = .scb
 567  01D8 F5            push   psw           ;save A (non-zero if error)
 568  01D9 CD0B02        call   mov7          ;restore hash info
 569  01DC 1EE0          mvi    e,usrcode     ;DE = .dcnt in scb
 570  01DE CD0B02        call   mov7          ;restore dcnt search addr & len
 571  01E1 2A7F03        lhld   udma
 572  01E4 EB            xchg
 573  01E5 CDA901        call   setdma        ;restore DMA to program's buffer
 574  01E8 AF            xra    a
 575  01E9 32FD02        sta    cbufp         ;reset buffer position to 0
 576  01EC F1            pop    psw
 577  01ED B7            ora    a
 578  01EE C9            ret                  ;zero flag set, if successful
 579                ;
 580                ;       reboot from ^C
 581                ;
 582                rebootx:
 583                       ;store 0fffeh in clp$errcode in SCB
 584  01EF 2AC602           lhld   scbadr
 585  01F2 2EAC             mvi    l,errflg
 586  01F4 36FE             mvi    m,0feh
 587  01F6 23               inx    h
 588  01F7 36FF             mvi    m,0ffh
 589  01F9 C30000           jmp    wboot
 590                ;
 591                ;
 592                ;       get input redirection mode to A
 593                ;       turn on ^C status mode for break
 594                ;       return .conmode+1 in HL
 595                ;       preserve registers BC and DE
 596                ;
 597                getmode:
 598  01FC 2AC602           lhld   scbadr
 599  01FF 2ECF             mvi    l,conmode
 600  0201 7E               mov    a,m
 601  0202 F601             ori    1             ;turn on ^C status
 602  0204 77               mov    m,a
 603  0205 23               inx    h
 604  0206 7E               mov    a,m
 605  0207 E603             ani    3             ;mask off redirection bits
 606  0209 3D               dcr    a             ;255=false, 0=conditional, 1=true,
 607  020A C9               ret                  ;  2=don't redirect input
 608                ;
 609                ;       move routine
 610                ;
 611  020B 0607     mov7:   mvi    b,7
 612                       ;                    HL = source
 613                       ;                    DE = destination
 614                       ;                    B = count
 615  020D 7E       move:   mov    a,m
 616  020E 12               stax   d
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
                        inx     h
                        inx     d
                        dcr     b
                        jnz     move
                        ret
                ;
                ;       add a to hl
                ;
0216 85         addhla: add     l
0217 6F                 mov     l,a
0218 D0                 rnc
0219 24                 inr     h
021A C9                 ret
                ;
                ;***************************************************************
                ;               BDOS CONSOLE INPUT ROUTINES
                ;***************************************************************

                ;
                ;       February 3, 1981
                ;
                ;
                ;       console handlers

0178 =          conin:  equ     coninf
                ;
                conech:
                        ;read character with echo
021B CD7801CD28         call conin! call echoc! rc      ;echo character?
                         ;character must be echoed before return
0222 F5CD5C02F1         push psw! call conout! pop psw
0227 C9                 ret                             ;with character in A
                ;
                echoc:
                        ;are we in cooked or raw mode?
0228 21D9023534         lxi h,cooked! dcr m! inr m! rz  ;return if raw
                        ;echo character if graphic
                        ;cr, lf, tab, or backspace
022E FE0DC8             cpi cr! rz              ;carriage return?
0231 FE0AC8             cpi lf! rz              ;line feed?
0234 FE09C8             cpi tab! rz             ;tab?
0237 FE08C8             cpi ctlh! rz            ;backspace?
023A FE20C9             cpi ' '! ret            ;carry set if not graphic
                ;
                conbrk:         ;STATUS - check for character ready
023D 21C302             lxi h,statflg
0240 4636FF             mov b,m! mvi m,0ffh     ;set conditional status flag true
0243 CDFC01             call getmode            ;check input redirection status mode
0246 FE01C8             cp 1! rz                ;actual status mode => return true
0249 B7C8               ora a! rz               ;false status mode  => return false
                        ;conditional status mode => false unless prev func was status
024B 78C9               mov a,b! ret            ; return false if statflg false
                                                ; return true if statflg true
                ;
                ;
                ctlout:
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
                        ;send character in A with possible preceding up-arrow
024D CD2802             call echoc              ;cy if not graphic (or special case)
0250 D25C02             jnc conout              ;skip if graphic, tab, cr, lf, or ctlh
                                ;send preceding up arrow
0253 F53E5ECD5C                 push psw! mvi a,ctl! call conout ;up arrow
0259 F1F640                     pop psw! ori 40h        ;becomes graphic letter
                                ;(drop through to conout)
                ;
                ;
                ;       send character in A to console
                ;
                conout:
025C 5F                 mov     e,a
025D 3AD802             lda     echo
0260 B7                 ora     a
0261 C8                 rz
0262 0E02               mvi     c,coutf
0264 C30900             jmp     next
                ;
                ;
                read:   ;read to buffer address (max length, current length, buffer)
0267 EB                 xchg                                    ;buffer address to HL
0268 4E23E50600         mov c,m! inx h! push h! mvi b,0         ;save .(current length)
                        ;B = current buffer length,
                        ;C = maximum buffer length,
                        ;HL= next to fill - 1
                        readnx:
                                ;read next character, BC, HL active
026D C5E5                       push b! push h                          ;blen, cmax, HL saved
                                readn0:
026F CD7801                             call conin                      ;next char in A
0272 E1C1                               pop h! pop b                    ;reactivate counters
0274 FE1AC28002                         cpi ctlz! jnz noteof            ;end of file?
0279 0504CAB902                         dcr b! inr b! jz readen         ;skip if buffer empty
027E 3E0D                               mvi a,cr                        ;otherwise return
                                noteof:
0280 FE0DCAB902                         cpi cr!   jz readen             ;end of line?
0285 FE0ACAB902                         cpi lf!   jz readen             ;also end of line
028A FE10C29F02                         cpi ctlp! jnz notp              ;skip if not ctlp
                                        ;list toggle - change parity
028F E5C5                               push h! push b                  ;save counters
0291 2AC6022ED4                         lhld scbadr! mvi l,listcp       ;hl =.listcp
0296 3E0196                             mvi a,1! sub m                  ;True-listcp
0299 77                                 mov m,a                         ;listcp = not listcp
029A C1E1C36D02                         pop b! pop h! jmp readnx        ;for another char
                                notp:
                                        ;not a ctlp
                                        ;place into buffer
                                rdecho:
029F 2377                               inx h! mov m,a          ;character filled to mem
02A1 04                                 inr b                   ;blen = blen + 1
                                rdech1:
                                        ;look for a random control character
02A2 C5E5                               push b! push h          ;active values saved
02A4 CD4D02                             call ctlout             ;may be up-arrow C
02A7 E1C17E                             pop h! pop b! mov a,m   ;recall char
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950
SER. #_____

```
02AA FE03                       cpi ctlc            ;set flags for reboot test
02AC 78                         mov a,b             ;move length to A
02AD C2B502                     jnz notc            ;skip if not a control c
02B0 FE01                       cpi 1               ;control C, must be length 1
02B2 CAEF01                     jz rebootx          ;reboot if blen = 1
                                ;length not one, so skip reboot
                        notc:
                                ;not reboot, are we at end of buffer?
02B5 B9DA6D02                   cmp c! jc readnx     ;go for another if not
                        readen:
                                ;end of read operation, store blen
02B9 E170                       pop h! mov m,b       ;M(current len) = B
02BB F5                         push psw            ;may be a ctl-z
02BC 3E0DCD5C02                 mvi a,cr! call conout ;return carriage
02C1 F1                         pop psw             ;restore character
02C2 C9                         ret
                ;
021B =          func1: equ      conech
                        ;return console character with echo
                ;
                ;func6:         see intercept routine at front of module
                ;
0267 =          func10:         equ     read
                        ;read a buffered console line
                ;
023D =          func11: equ     conbrk
                        ;check console status
                ;
                ;

                ;*****************************************************************
                ;               DATA AREA
                ;*****************************************************************

02C3 00         statflg:        db      0       ;non-zero if prev funct was status
                        ;
                        ;

                ;*****************************************************************
                ;       Following variables and entry points are used by GET.COM
                ;       Their order and contents must not be changed without also
                ;       changing GET.COM.
                ;*****************************************************************
                        ;
                        if bios$functions
                        ;
                exit$table:                     ;addresses to go to on exit
02C4 0900               dw      next            ;BDOS
                        endif
                        ;
                movstart:
                init$table:                     ;addresses used by GET.COM for
02C6 0E00       scbadr:         dw      kill            ;address of System Control Block
                        ;
                        if bios$functions       ;GET.RSX initialization
                        ;
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

```
02C8 3800       biosta dw    bios$constat   ;set to real BIOS routine
02CA 3E00       biosin dw    bios$conin     ;set to real BIOS routine
                       ;
                                           ;restore only if changed when removed.
                restore$mode
02CC 00                db    0              ;if non-zero change LXI @jmpadr to JMP
                                           ;when removed.
                restore$bios:
                       ;hl = real constat routine
                       ;de = real conin routine
02CD 220000            shld  0              ;address of const jmp initialized by CDM
02D0 EB                xchg
02D1 220000            shld  0              ;address of conin jmp initialized by CDM
02D4 C9                ret
                       endif
                       ;
                real$bdos:
02D5 C30500            jmp   bdos           ;address filled in by CDM
                       ;
                       ;
02D8 01         echo:  db    1
02D9 00         cooked:       db    0
                       ;
                program:
02DA 00                db    0              ;true if program input only
02DB 00         subusr:       db    0              ;user number for redirection file
02DC 01         subfcb:       db    1              ;a:
02DD 535953494E        db    'SYSIN   '
02E5 535542            db    'SUB'
02E8 0000              db    0,0
02EA 00         submod:       db    0
02EB            subrc: ds    1
02EC                   ds    16             ;map
02FC            subcr: ds    1
                       ;
                movend:
                ;*****************************************************************

02FD 80         cbufp  db    128            ;current character position in cbuf
02FE 0D         nextchr       db    cr             ;next character (1 char lookahead)

                       if submit
                ccp$line:
                       db    false          ;nonzero if line is for CCP
                       endif

                cbuf:                       ;128 byte record buffer

02FF 0303030303        db    3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3
030F 0303030303        db    3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3
031F 0303030303        db    3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3
032F 0303030303        db    3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3

033F 0303030303        db    3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3
034F 0303030303        db    3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3
035F 0303030303        db    3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
 841  036F 0303030303          db      3,3,3,3, 3,3,3,3, 3,3,3,3, 3,3,3,3
 842
 843  037F 8000       udma:    dw      buf             ;user dma address
 844                  get$active:
 845  0381 81                  db      gkillf
 846                           ;
 847                  sav$area:                       ;14 byte save area (searchn)
 848  0382 6868686868          db      68h,68h,68h,68h,68h, 68h,68h,68h,68h,68h
 849  038C 68686868            db      68h,68h,68h,68h
 850  0390 00         excess:  db      0
 851                  old$stack:
 852  0391 0000                dw      0
 853                           if      submit
 854                  garbage:
 855                  ;        db      cr,lf
 856                           db      'WARNING: PROGRAM INPUT IGNORED',cr,lf,'$'
 857                           else
 858  0393 01         ccpcnt:  db      1
 859                           endif
 860                  patch$area:
 861  0394                     ds      30h
 862  03C4 2031353132          db      ' 151282 '
 863  03CC 20434F5059          db      ' COPYR ''82 DRI '
 864  03DB 6767676767          db      67h,67h,67h,67h,67h, 67h,67h,67h,67h,67h
 865  03E5 6767676767          db      67h,67h,67h,67h,67h, 67h,67h,67h,67h,67h
 866  03EF 6767676767          db      67h,67h,67h,67h,67h, 67h,67h,67h,67h,67h
 867                           ;
 868                  stack:                          ;15 level stack
 869  03F9                     end
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

ADDHLA 0216 236 506 625#
BDOS 0005 48# 49 131 802
BDOSL 0006 49# 291 297
BDOSOFF 00FE 109# 299
BIOSCONIN 003E 179# 786
BIOSCONSTAT 0038 170# 785
BIOSFUNCTIONS FFFF 43# 147 172 181 229 773 783
BIOSIN 02CA 153 786#
BIOSTA 02C8 155 785#
BIOSTRAP 004C 176 186 197#
BREAK 0145 409 443#
BUF 0080 50# 843
CBUF 02FF 505 547 831#
CBUFP 02FD 498 504 575 823#
CCPCNT 0393 390 858#
CCPFLG 00B3 107# 286 347 384
CCPRES 0020 108# 351 388
CINF 0001 75# 214 421
CKKILL 00AB 259 265#
CKSIG 009F 257# 393
CLOSEF 0010 83# 277
CONBRK 023D 433 487 661# 754
CONECH 021B 643# 746
CONIN 0178 641# 645 703
CONINF 0178 359 437 495# 641
CONINFX 0074 119# 162
CONMODE 00CF 113# 599
CONOUT 025C 647 675 677 684# 742
CONSTFX 006E 118# 159
COOKED 02D9 652 806#
COUTF 0002 76# 689
CR 000D 69# 517 535 655 707 709 742 824 856
CRAWF 0006 77# 184 210 451
CREADF 000A 78# 216 419
CSTATF 0008 79# 175 208 423 445
CTL 005E 71# 677
CTLC 0003 54# 729
CTLE 0005 55#
CTLH 0008 56# 658
CTLOUT 024D 672# 727
CTLP 0010 57# 711
CTLR 0012 58#
CTLS 0013 59#
CTLU 0015 60#
CTLX 0018 61#
CTLZ 001A 63# 65# 473 502 511 705
DCNT 00E1 117#
DELF 0013 84# 279
DIRINP 0173 431 486#
DMAAD 00D8 115#
DMAF 001A 86# 220 540
DREADF 0014 85# 545
ECHO 02D8 686 805#
ECHOC 0228 645 650# 674
ERRFLG 00AC 110# 585
EXCESS 0390 194 360 850#

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950
SER. #

```
EXIT          007E  204   221   226#  252   267   326   468
EXITTABLE     02C4  235   775#
FALSE         0000   12#   14    15   828
FUNC1         021B  422   746#
FUNC10        0267  420   751#
FUNC11        023D  424   754#
FUNC6         0130  426#
GETACTIVE     0381  260   271   844#
GETC          0178  496#  522   531
GETC1         0190  503   510#
GETMODE       01FC  336   457   597#  664
GETOUT        001B  145#  274
GFCBF         0082   93#  254
GINITF        0080   91#  248
GKILLF        0081   92#  261   266   845
GSIGF         008C   99#  258
INITTABLE     02C6  247   780#
INTERCEPT     00D3  209   211   215   217   322#
JFCBF         008F  102#
JINITF        008D  100#
JKILLF        008E  101#
KILL          000E  132#  250   282   324   368   781
LF            000A   70#  521   656   710   856
LISTCP        00D4  114#  714
LOADF         003B   89#  311
MOV7          020B  554   556   568   570   611#
MOVE          020D  615#  620
MOVEND        02FD  820#
MOVSTART      02C6  779#
NBANK         000F  133#
NEXT          0009  130#  231   296   313   366   395   453   541   564   690
                    776
NEXTCHR       02FE  515   824#
NOTC          02B5  731   735#
NOTEOF        0280  705   708#
NOTP          029F  711   718#
OLDSTACK      0391  331   466   475   851#
OPENF         000F   82#
PATCH         001A  136#
PATCHAREA     0394  860#
PBUFF         0009   81#  365
PCFCBF        0086   96#
PCHRF         0005   80#
PCKILLF       0085   95#
PGDEF         00C9  112#
PGMODE        00C8  111#  345   401
PINITF        0084   94#
PLFCBF        008A   98#
PLKILLF       0089   97#
PREV          000C  131#
PROGRAM       02DA  382   808#
RDECH1        02A2  724#
RDECHO        029F  721#
READ          0267  693#  751
READEN        02B9  706   709   710   738#
READF         01AE  500   545#
```

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

READNO	026F	702#						
READNX	026D	699#	717	737				
REALBDOS	02D5	446	801#					
REBOOTX	01EF	582#	733					
REDIRECT	0115	356	389	407#				
REMOVERSX	0000	15#	294					
RESTOR	00B0	270#	392	450	513			
RESTOREBIOS	02CD	156	792#					
RESTOREMODE	02CC	150	789#					
RETMON	0168	415	471#					
RNAME	0010	134#						
RSXF	003C	90#	218	394				
RSXFUNC	008A	219	245#					
RUBOUT	007F	67#						
SAVAREA	0382	550	847#					
SCBA	003A	106#						
SCBADR	02C6	158	285	549	584	598	714	781#
SCBF	0031	88#						
SERIAL	0000	126#						
SETDMA	01A9	540#	548	573				
SETNOPAGE	010F	386	398#					
SKIP	0157	338	379	396	455#	474		
SKIP1	0161	441	466#					
SPACE	0018	135#						
STACK	03F9	332	868#					
STATFLG	02C3	212	434	662	763#			
SUBCR	02FC	818#						
SUBDOS	01B0	278	280	546#				
SUBFCB	02DC	255	811#					
SUBMIT	0000	14#	62	276	340	526	826	853
SUBMOD	02EA	815#						
SUBRC	02EB	816#						
SUBUSR	02DB	559	810#					
TAB	0009	68#	657					
TRAP	0046	129	191#					
TRAPJMP	0006	128#	202					
TRUE	FFFF	11#	43	363				
UDMA	037F	223	571	843#				
USERF	0020	87#						
USRCODE	00E0	116#	555	560	569			
WBOOT	0000	47#	374	589				

COPYRIGHT © 1982
DIGITAL RESEARCH
P.O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____

COPYRIGHT © 1982
DIGITAL RESEARCH
P. O. BOX 579
PACIFIC GROVE, CA 93950

SER. #_____